



(12)

(43) Date of publication:

(51) Int Cl.⁷: **G06F 17/60**

(21) Application number: 00402519.3

(22) Date of filing: 13.09.2000

(84) Designated Contracting States:

AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU
MC NL PT SE

Designated Extension States:

AL LT LV MK RO SI

(71) Applicant: iMediation, S.A.

92045 Paris La Defense (FR)

(72) Inventors:

- **Baudu, Regis Jacques**
75018 Paris (FR)

- **Colin, Dominique Michel**
78810 Feucherolles (FR)

- **Cores, Andres**
75016 Paris (FR)

- LHospitalier, Denis Rene Marie
75012 Paris (FR)

- **Sair, Faraj**
92250 La Garenne Colombes (FR)

(74) Representative: **Harris, Ian Richard**

D. Young & Co.,

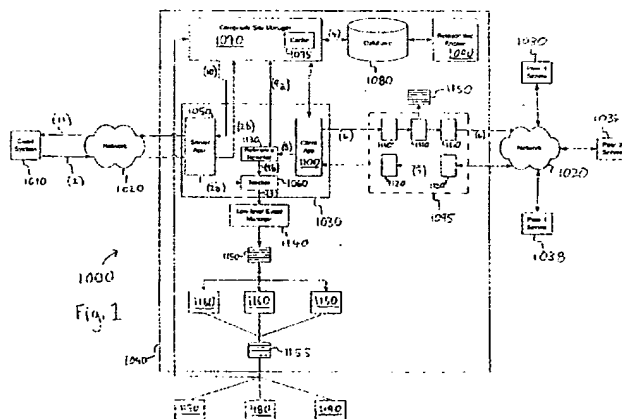
21 New Fetter Lane

London EC4A 1DA (GB)

(54) **A method and system for managing network-based partner relationships**

(57) Disclosed are computer-implemented systems and methods for the management and operation of network-based partner, e.g., business, relationships. A representation of the partner relationship is established that includes rules for allocation of compensation, the subject matter to come under the partner relationship, and a resources to be presented to users/customers when they encounter an aspect of the partner relationship, for instance by browsing a web site of one of the partner entities. A system implementing the partner relationship includes a transactional engine for monitoring the browsing activity of a user while browsing partner relationship resources. A composite site manager gener-

ates the partner relationship resources based on component resources drawn from the partner entities and a presentation characterization forming part of the relationship representation. The transactional engine generates event data from the client-server request/response data flow. Message brokers transport the event data to software component that accumulates and aggregates the event data to generate facts that are meaningful for the operation of the partner relationship, e.g., compensation flows. An interposition mechanism allows for the request/response data flow to be modified. Further, the interposition mechanism can be used to implement common wallet or common shopping basket functionality in the context of the partner relationship.



Description**FIELD**

- 5 [0001] Features of the invention relate generally to computer-implemented systems for the setup and management of commercial relationships in a network-based environment, and more particularly to systems achieving improved deployment, flexibility and scalability.

BACKGROUND

- 10 [0002] Increasingly, businesses are conducting their operations over public computer networks, most prominently the worldwide web aspect of the internet. These open networks provide profitable opportunities for groups of entities to form partnerships. Indeed, the enormous scope and variety of electronic commerce alliances formed in recent years among trading partners using the worldwide web bears witness to the desirability of such alliances. Both from the
 15 perspective of consumers, who are better able to find goods and services of interest, and from the perspective of sellers who are better able to expand their market through alliances, successful implementation of network-based partnering relationships is highly desirable.

- [0003] However, software solutions for implementing such relationships leave deficiencies in effective implementation and management of network-based partnering relationships. One common example is the situation where a merchant selling goods through an on-line catalog desires to establish an affiliate network to drive traffic to the merchant site. In exchange for driving a purchaser to the merchant site, the affiliate could receive compensation, say a commission of purchases.

- [0004] From the perspective of the merchant, establishing the mechanisms on their site to track affiliate-generated traffic, and provide compensation can be costly and time consuming, and may involve significant intrusive re-editing of their on-line catalog. From the perspective of the affiliate, there is an information deficiency: what is the real value of the traffic being driven to the merchant and how can the compensation reflect this value. Still further, from the perspective of the merchant-affiliate partnership it would be desirable for a solution to exist that allowed for co-marketing, for instance with co-branding, so that both partners could reap the benefits of the partnership. Accordingly, there is a need for system that allows merchants to conveniently implement an affiliate network, that provides robust
 30 information generation, and a convenient capability of providing partnership-specific content presentations.

- [0005] Another common example is for a portal-type site. A portal-operator typically drives traffic to many merchants without having a means to capture this value. Indeed, a portal operator may desire to not completely drive user-traffic to merchant sites when a purchase is to be made, but rather, have the purchase made via the portal site thereby keeping the user. Many portals have high traffic rates and, in such a situation, effective scalability is highly desirable. Still further, from the perspective of the portal-operator it would be desirable for a solution to exist that allowed features such as a common shopping basket or common wallet to exist for all or selected merchants accessible through the portal. Thus it would be desirable for methods to exist that allowed for a portal-operator to implement a purchasing system that allowed it to manage purchases made on merchant sites by users coming to the merchant through the operator's portal. Still further, it there is a need for such methods to be highly scalable, flexible, and extensible to enable
 40 the portal-operator to coordinate with the various purchasing systems of the merchants.

- [0006] Yet another common network-based partnership scenario is for a reseller network. A manufacture with a well-known brand, may wish to direct user-traffic to a local reseller for purchase, follow-up service, localization, special promotions etc. However, simply redirecting users to the local reseller deprives the manufacturer of information about the efficacy of its reseller or distribution network. There is a need for a software system to exist that allows for manufacturer in this situation to track information after the user has been directed to their local reseller. Still further, the entire reseller network could benefit from traffic, sales, and other information about all members of the reseller network. Thus there is a need for an efficient information dissemination mechanism in connection with network-based partnering relationships.

- [0007] There are conventional systems which provide benefits in connection with network-based partnering relationships, but leave several deficiencies. One related means are OLAP and other data mining tools. These tools allow web-site operators to examine and analyze their server logs and from this gain certain forms of information. However, such tools do not assist in implementing a network-based partnering relationship in the first instance, only in gathering data after implementation. Still further, such tools do not provide real-time or near-real-time data but rather are typically based on batch runs through voluminous server logs. This type of delay is undesirable in a rapidly-moving market where commissions could be calculated daily. More fundamentally, this low-level information is not useful in the context of the network-based partnering relationship and must be mapped to the higher-level semantic of the business relationship.

- [0008] Another related conventional technique are server-side dynamic web page generation tools, e.g. Active Server

Pages, Java Server Pages, PHP, or conventional CGI-based scripts/programs. These systems do provide for a general means to dynamically create web pages, however do not provide a convenient and open means to establish commercial partnerships, let alone effectively track events that occur in the context of those partnerships. Still further, they provide no effective means to disseminate information to several peer entities in a network-based partner relationship.

[0009] Yet another set of related solutions are so-called "link-sharing" or associates-linking systems. These systems typically provide software for administering an affiliate relationship. They frequently require substantial deployment time and effort, with both the merchant and affiliates having to modify their sites. Further, typical solutions of this type are limited to the connecting link between the merchant and the affiliate and are an inadequate solution to gain full tracking information of the user's browsing session in the context of the partner relationship. To gain this type of information in the context of using such tools typically requires reversion to server-log or other data-mining techniques which, as noted above, scale poorly and do not provide real-time data.

[0010] Accordingly, there is a need for a methods and systems for implementing and managing network-based partnering relationships that is convenient to deploy, highly scalable, provides for comprehensive real-time information gathering, generation of higher-level semantic information, and efficient dissemination of information.

SUMMARY

[0011] Particular and preferred aspects of the invention are set out in the accompanying independent and dependent claims. Features of the dependent claims may be combined with those of the independent claims as appropriate and in combinations other than those explicitly set out in the claims.

[0012] These and other benefits are obtained by the present invention that provides methods and systems for managing network-based partner relationships. One aspect of the invention provides a computer-controlled method of operating a network-based partner relationship. An illustrative method includes establishing a representation of a relationship between two peer entities. The representation includes collection of resources associated with the relationship, and a presentation template to be applied when providing the collection of resources. The representation also includes criteria for providing a compensation flow between the two peer entities. An additional part of the illustrative method includes dynamically generating composite resources during a browsing session. The composite resources include first and second components selected from the collection of resources associated with the first and second peer entities. This method also includes monitoring the browsing session for generating session events and transforming the session events to events in a semantic associated with the relationship, thus generating higher-level semantic events. Then, a compensation flow is generated based on the higher-level semantic events based on the criteria in the relationship representation.

[0013] In a variation of this illustrative method, generating session events also includes providing the session events on a first message broker, such as Java Messaging Services, then transforming the session events includes receiving the session events from the message broker and providing the higher-level semantic events on a second message broker. An additional aspect in this variation is where dynamically generating composite resources during a browsing session comprises includes receiving the higher-level semantic events from the second message broker and selecting one or more of the components based on the higher-level semantic events.

[0014] In yet another variation, dynamically generating composite resources during a browsing session may include initiating a request for a component resident on target site, receiving and altering the request, and then forwarding the altered request to the target site. In an additional aspect, information may be extracted from the request, for instance for implementing a common shopping basket or common wallet. Then, the extracted information could be used to suitably update the data implementing the common wallet/shopping basket.

[0015] In a related variation, dynamically generating composite resources during a browsing session includes initiating a request for a component resident at a target site, receiving and altering a response from the target site, and forwarding the altered response.

[0016] Yet another aspect of the invention are computer controlled network based partnership systems. An illustrative system includes a relationship engine for defining attributes of a relationship among at least two peer entities and a storage for storing the attributes of the relationship. The illustrative system also includes a composite site manager for dynamically generating composite resources during browsing sessions. The composite resources include component resources associated with each of the two peer entities and the composite site manager is configured for retrieving a characterization of the composite resources from the storage. This system also includes a session tracking component. The session tracking component includes a server application for receiving requests from a client system, a communication client configured for requesting the composite resources from the composite site manager, a client application configured to request the component resources from server systems associated with the two peer entities and a session event generation module configured for generating event data respecting the requests and the component resources. This system also includes a set of semantic mapping modules that are configured for receiving the event data. The semantic mapping modules include rules for generating higher-level semantic events based on the event data.

[0017] A variation of illustrative system just described also includes a message broker. The message broker is configured for receiving the higher-level semantic events from the set of semantic mapping modules and the composite site manager is further configured for receiving the higher level semantic events from the message broker. In this variation, the composite site manager dynamically generates the composite resources responsive to the higher-level semantic events.

[0018] Another variation also includes an interposition module logically positioned between the client application and the server systems. The interposition module is configured for altering the request sent by said client application for the component resources.

[0019] In a related variation, the illustrative system also includes an interposition module logically positioned between the client application and the server systems. The interposition module is configured for altering the component resources sent by the server systems.

[0020] To introduce features of the invention more fully set forth below, a summary of an illustrative embodiment will now be described. A web-based merchant seeks to establish an affiliate network. The merchant deploys a software package embodying features of the invention on a computer system (the "platform"). To attract affiliates the merchant uses a web-based interface to set up a proposal for potential affiliates (an "offer"). The offer includes a template of a "composite site" that will be used to present co-branded pages to users browsing the merchant's catalog via the affiliate's site. The offer also includes portions of the merchant's catalog coming under the scope of the offer, as well as compensation rules, for instance, a commission rate based on sales volume. The merchant makes the offer publicly available on the worldwide web. Interested potential affiliates review the offer and indicate interest. For those the merchant accepts, a partner relationship is thus established and the platform provides the affiliate with a link to place on their site with an identifier of the composite site. When user follows this link they are directed to the platform system. The platform dynamically generates pages based on the composite site definition. The composite site pages include components from the merchant's catalogs as well as brand images and documents from both the merchant and affiliate. When the platform retrieves these components it rewrites links in the components to point back to the platform. Thus, the platform acts as a mobile proxy through which the user's browsing session is controlled. The platform tracks the browsing session and generates event data from the session including, for instance, when the user purchases an item from the merchant's catalog. This event data is passed to a series of modules that take this low session-level data, aggregate it, and generate facts in a semantic of a higher level—the level of the business relationship between the merchant and affiliate. These higher level events are provided on a flexible messaging architecture so that, not only the partners can effectively track their relationship, but also that this information can be provided to other affiliates in a flexible and scalable manner. The platform also tracks these higher-level events and computes commissions due based on the purchases. The merchant and/or affiliate can then access a web site for monitoring the details of their partnership, e.g., commission earned or other statistics.

BRIEF DESCRIPTION OF THE DRAWINGS

[0021] The above features and advantages will be better appreciated with reference to the following detailed description and identified figures where, in accordance with illustrative embodiments:

Fig. 1 illustrates features of a system for managing networked partner relationships;

Fig. 2-1, 2-2, and 2-3 depict functions available to various peer entity relationship participants;

Fig. 3 depicts a state diagram illustrating a composite site browsing session;

Fig. 4 depicts schematic of a composite site page as it could be parsed and rendered on a user's browser; and

Fig. 5 depicts a block diagram of a reference rewriting module.

DETAILED DESCRIPTION

DESCRIPTION OF FIGURES

[0022] Fig. 1 illustrates features of a system for managing networked partner relationships 1000 in accordance with an illustrative embodiment. A client system 1010, under control of a user, initiates a request (1) across a network 1020 to a host computer system executing software implementing the system 1000 indicated in Fig. 1 as a platform 1040.

[0023] In some embodiments, the client system 1010 is a conventional "web" browser, configured for operation with either HTML and/or XML executing on general purpose computer. In other embodiments, the client system 1010 could

be special-purpose computing hardware, including, for instance, a mobile phone, personal digital assistant, or set-top box. Given its current commercial ubiquity, in preferred embodiments, the network 1020 includes a portion of the Internet; however this is not fundamental and other networks, either public or private, and using either the TCP/IP stack of protocols or other protocols could be used.

5 **[0024]** A server application 1050 receives the request from the client system 1010. The request preferably includes a plurality of field/value pairs: an identifier of a 'composite site' (csid), a session identifier (sid), a customer identifier (cid), a merchant identifier (mid), a partner id (rid), an encoded HTTP referer, and a target frame in which to load the response. These particular fields are not fundamental. In some embodiments, the field/value pairs are part of the path or data portion of a URL, in others they could be passed by cookies, hidden fields, or other means of client-server data
10 transfer available to one skilled in the art. The server application 1050 parses and extracts the field values from the request and passes events (2a) from the user's browsing session to a tracker module 1060. The tracker module 1060 is explained in greater detail below in connection with Fig. 4; briefly, here, certain events from the user's browsing session are tracked for use in generating higher-level semantic events. In the illustrative embodiment, the conditions that must be satisfied to trigger the firing of rules that generate the higher-level semantic events determine which events
15 are tracked. Features of this aspect could be carried on as illustrated in a patent application, filed concurrently herewith for the same applicant, entitled "Method and System for Transforming Session Data,".

[0025] The server application 1050 passes the composite site identifier (2b) to a composite site manager 1070. The composite site manager 1070 embodies features described in a patent application filed concurrently herewith and incorporated herein by this reference, entitled Composite Site Generation System. In some embodiments, the composite site manager 1070 includes software components configured for generating web pages, e.g., HTML documents, or XML documents with an XSL style sheet for presenting the XML data. The collection of web pages provided through the composite site manager 1070 to the user of the client system 1010 constitute the composite site. For ease of exposition, a single document or web page forming a portion of the composite site is also referred to herein as the composite site or a composite site resource.
20

25 **[0026]** The site is 'composite' in that the web pages themselves comprise component resources requested from target sites, such as a first peer server 1038 and a second peer server, 1034 that are assembled by the composite site manager 1070. The assembling of the component resources is determined by a definition of the composite site. These features are further illustrated below with reference to Fig. 4. Additionally, further related detail can be found in the disclosure of a concurrently filed patent application entitled "Method and System for Composite Site Resource Generation" set forth as **ANNEX A** hereto.
30

[0027] The composite site manager 1070 uses the CSID to retrieve (4) the definition of the composite site from a database 1080. In some embodiments, the definition of the composite site is stored in the database 1080 in conventional relational tables and mapped to an XML schema upon retrieval (4). The composite site manager 1070 may store the composite site definition in a cache 1075, thereby reducing database query latency when repeatedly generated the composite site resources.
35

[0028] The composite site manager 1070 parses the composite site definition to identify component resources that should be retrieved to assemble the composite site. The composite site manager 1070 instructs a client application 1100 to request (6) component resources resident on remote systems.

40 **[0029]** Some embodiments of the invention include an interposition mechanism. The interposition mechanism can be used to perform actions on for instance, HTTP, Request and Response messages. The interposition mechanism can produce events relative to the interaction between the platform 1040 and peer server systems (described below). The interposition mechanism further can perform lightweight HTTP flow updating and/or redirection. The interposition mechanism achieves these features by intercepting and modifying, for instance, HTTP Request / Response flow via modules. An interposition engine 1095 operates with and controls the sequence of execution of the modules. Interposition modules perform the actual functional modification of the HTTP flow. In some embodiments, one or more filters are used to determine whether a particular interposition module has to perform an action or not. One skilled in the field will appreciate that such filters can avoid having the modules perform generic manipulation. One or more dispatchers, which are abstract components, handle the filters. Typically a dispatcher is followed by one or more filters, which, in turn, are followed by one or more modules. When a request or response enters the interposition engine 1095, all filters
45 are executed on it to determine which (if any) modules meet the criteria of the filters. Preferably, modules execute only after all filters have executed.
50

[0030] In an illustrative embodiment, one or more first interposition modules 1100 may alter the request (6). In accordance with this embodiment, an interposition module is a software module that examines a locator in the request (6), e.g. a URL, and if a particular pattern is satisfied, the interposition module executes code to perform an action
55 either on the locator itself, or more generally on the state of the platform 1040. For instance, in some embodiments interposition modules could be used to implement a common 'shopping basket' or a "common wallet" on an electronic commerce portal. Yet another aspect of the invention are methods using interposition modules to transform data flow in a browsing session. The transformed data flow may be to provide an electronic commerce transaction function in

the context of the network-based partner relationship.

[0031] In some embodiments, one of the component resources includes a page from a merchant's on-line catalog. Peer entities may be merchants with such on-line catalog. For instance, a first peer entity may operate a first peer server system 1038 through which the peer entity operates an electronic commerce operation including an on-line catalog of items for which users can make purchase orders. A page forming a portion of this on-line catalog can constitute one of the component resources. Similarly, other component resources could be requested from a second peer server system 1034 operated by a second peer entity distinct from the first peer entity. More generally, the component resources could include any addressable resource capable of being parsed and rendered by the client system 1010.

[0032] A response (7) including the component resource requested in the request (6) returns. One or more second interposition modules 1120 may alter the response (7) as the first interposition modules 1110 may alter the request (6). The client application 1100 receives the response (7) which is provided (8) to a reference rewriting module 1030.

[0033] The reference rewriting module 1130 rewrites certain resource references in the response (7) to ensure that requests for these resource made by the client system 1010 are directed to the platform 1040. This is described in greater detail below in connection with Fig. 5. Briefly here, references to resources including, for instance, URLs, are rewritten from identifying a resource accessible from a target site such as the first peer server system 1038 to a resource accessible from the platform 1040. This step thus provides a facility for the platform 1040 to track and monitor a browsing session of the client system 1010 as more fully described below.

[0034] The reference rewriting module 1130 provides (9a) the component resources (with resource references appropriately rewritten) to the composite site manager 1070 that assembles the component resources in accordance with the definition of the composite site to generate a composite site resource. The composite site manager 1070 provides (10) the resulting resource to the server application that then provides this as a response (11) to the request (1) from the client system 1010.

[0035] The server application 1050, the reference rewriter 1130, tracker module 1060 and client application 1100 form a transactional engine 1030 for tracking the browsing session and creating session-level semantic events. In some embodiments, the transactional engine 1130 is multithreaded with its various components communicating through shared memory. In some embodiments it is implemented in a language such as C for increased efficiency.

[0036] During the above-described process events are detected and reported from the data flow. In part, this is described above in connection with the server application 1050 passing events (2a) to the tracker module 1060. In addition, the reference rewriting module 1130 may be configured to detect and report events 9b based on the contents of the component resource when parsing the component resources to detect references for rewriting. For instance, if the component resource were a page from a merchant's on-line catalog that indicated a price for an item that was purchased, the price could be detected and reported as the event 9b. As one skilled in art having the benefit of this disclosure will appreciate, the type and particulars of events 9b detected and reported in this fashion is limited only by the data flow in the browsing session.

[0037] One deficiency, however, in the data flow of a client-server request/response cycle is that it is in the semantic of the request/response cycle. Ordinarily this semantic level is too low for purposes of effectively managing network partner relationships. Fundamentally, any fact in a semantic that spans multiple requests/responses cannot be represented by an event drawn from a single request/response cycle. Yet, it is in such higher-level semantics that facts useful for the management of the partnership relationship are desired. The session-level semantic event that a document was received that included a particular number in a particular place in a particular pattern is too low to take meaningful action in a partner relationship, e.g., crediting a commission from a merchant to an affiliate. In a higher-level semantic, the document has the meaning of an order confirmation page and the number is the purchase price. In some embodiments, events in the session-level semantic are extracted from HTTP Request and Response messages; these events are accumulated or aggregated, and higher-level semantic events are derived through a rule-based system where activation of a rule coincides with transformation to the higher-level semantic that can draw on the aggregated or accumulated information.

[0038] A mechanism for generating higher-level semantic events from the low session-level semantic events involves the tracker module 1060 providing (3) session-level semantic events (2a & 9b) to a low level event manager 1140. A method and system for performing this feature are described in the aforementioned patent application entitled "Method and System for Transforming Session Data," filed concurrently herewith forming Annex B.

[0039] The low level event manager 1140 publishes a sequence of tracked session level semantic events across a first message broker 1150. In some embodiments the message broker uses Java Messaging Services, although other systems could be used. A set of higher-level semantic event generators 1160 subscribe to the first message broker 1150 and generate events in a higher-level semantic from the events in a session-level semantic. This is described in greater detail in the concurrently filed patent application entitled "Method and System for Transforming Session Data". Note also that an interposition module, such as one of the first interposition modules 1110 may also publish events on the first message broker 1150 either directly as illustrated in Fig. 1, or by providing them upstream of the message broker 1150 to the low-level event manager 1140.

[0040] The set of higher-level semantic event generators **1160** provide higher-level semantic events across a second message broker **1155**. The second message broker **1155** may be of the same or different type of messaging service as the first message broker **1150**. Also, in some embodiments the second message broker **1155** publishes across a public network.

[0041] Applications for managing aspects of the network partner relationship can subscribe to the higher-level semantic events provided on the second message broker **1155**. In some embodiments, a compensation management application **1170** subscribes and manages flows of compensations among the peer entities forming the network partner relationship. In some embodiments, a statistics generation application **1180** subscribes and generates summary data and analysis about financial, administrative, or network aspects of the network partner relationship. More generally, a custom application **1190** can be written to provide particular services based on the higher-level semantic events.

[0042] Still further, in some embodiments the composite site manager **1070** receives higher-level semantic events from the second message broker **1155** and dynamically alters aspects of composite site resources based on the higher-level semantic events.

[0043] Fig. 2-1, 2-2, and 2-3 depict functions available to various peer entity relationship participants in accordance with an illustrative embodiment. In some embodiments the functions depicted in these figures are accessed through a browser-based interface to the relationship engine **1090** where server-side logic controls generation of the pages implementing the interface on the browser and generates/updates the appropriate data objects in the database **1080** based on browser-user input.

[0044] Fig. 2-1 depicts functions available to a peer entity identified as a 'Merchant.' A create offer function **2110** involves creating a definition of a (potential) peer entity relationship ("offer"). Creating an offer involves identifying items covered by the offer **2150**. In some embodiments, the items are identified with reference to the on-line catalog of the Merchant although the items covered could be identified in other ways. Creating an offer also involves determining compensation and other terms **2130** for the offer.

[0045] In some embodiments the compensation terms are rule-based and provide conditions that must be satisfied and the resulting compensation if the conditions are satisfied. For instance, the Merchant may pay a 5% commission if an item is sold. One skilled in the art having the benefit of this disclosure, will readily appreciate that the particular rules determining compensation are not fundamental and may be tailored to the particular circumstances of the peer entities. Further, certain common compensation rules (e.g. commission as a percentage of sales price) could be predetermined and that other compensation rules for be custom-developed. In some embodiments, offers include other terms including, for instance, an exclusivity term, a compensation period, a renewal schedule, etc. As with the compensation rules, these could be predetermined and selected among or custom-developed.

[0046] Creating an offer also involves determining access rights **2140** to information generated in connection with implementation of the offer and privileges for changing aspects the offer. Finally, another aspect of creating an offer is selecting a composite site definition **2120**. The composite site definition determines a manner in which component resources from various peers, e.g., the Merchant, will be assembled during a user's browsing of resources within the scope of the offer.

[0047] Another function available to the merchant is viewing compensation or statistics **2170**. After an offer has been accepted (as described below) activity within the scope of the offer may generate compensation flow to or from the Merchant and may generate statistical information. The viewing compensation or statistics **2170** function allows the Merchant to have this information presented.

[0048] Yet another function available to the Merchant is to accept or decline potential peer entities **2160**. In particular, after an offer is created, interested entities may seek to accept the offer (as described below) and thus become peer entities. When such an entity has indicated a desire to accept the offer, the Merchant can either accept or decline.

[0049] After an offer has been created, it is made available and interested entities may review it and indicate their desire to accept the offer. Those entities accepting offers are termed "Affiliates". As depicted in Fig. 2-2, functions available to an Affiliate include an accept offer function **2100**. The accept offer function **2100** may also be accessed with a browser-based interface to server-side logic as described above in connection with the Merchant functions. In addition, the Affiliate defines the composite site **2110** that will be used with the offer. Using the definition of the composite site selected by the Merchant, the Affiliate specifies their component resources that will be used in presenting the composite site during the user's browsing session. As with the Merchant, the Affiliate may view compensation or statistics.

[0050] A third type of entity in addition to Merchant and Affiliate is termed an "Executive." The Executive has functions available to it in addition to those of the Merchant. In particular, the Executive identifies a currency **2135** that will be used for offers implemented by the platform. Finally, the Executive may accept or decline Merchants **2180** that may desire to create offers.

[0051] Fig. 3 depicts a state diagram illustrating a composite site browsing session **3000** in accordance with an illustrative embodiment in which a user operates a conventional web browser. Initially the user in an internet browsing state **3100** follows a link to a peer site **3150** and enters a peer browsing state **3200**. While in the peer browsing state

3200, for instance navigating a site of an Affiliate, the user follows link to a composite site **3250**. More particularly, the link to a composite site **3250** is a URL of the form:

<protocol>://<path to platform>/<field1>/<field value1> ... <fieldN>/<field valueN>/<id. of target resource>.

[0052] The portion <id. of target resource> is an identifier of a resource that will be included in the composite site resource provided to the user when a request is made for the resource associated with the link to a composite site **3250**. In some embodiments the identifier is a URL; in others a key used in a lookup data structure, and, in any event, the particular form is not fundamental. The link to a composite site **3250** is associated with a particular offer and is provided to the Affiliate in connection with offer creation/acceptance interaction previously described. That is, when a Merchant has agreed to an Affiliates' acceptance of an offer the link to a composite site **3250** is provided to the Affiliate to place on the Affiliate's site. Thereafter, when a user follows this link, their session is tracked through the platform **1040**.

[0053] Now in a composite site browsing state **3300**, the resources provided to the user's client system have two features. First, when following a link to request a resource, the response sent to the user's browser is an composition of component resources (as is further described below in connection with **Fig. 4**) including resources from the Merchant peer entity and the Affiliate peer entity. Second, links in the component resources (with a few exceptions described below) are rewritten to point to a server system hosting the platform **1040** creating the composite site. For instance, one of the component resources in a composite site page could be a page from an on-line catalog of the Merchant. This page could initially have a link that points to a different page in the Merchant's on-line catalog. This link is rewritten to point to the platform **1040** and to include an identifier of the different page.

[0054] One skilled in the art will appreciate that while following links provided as part of the composite site, the user's browsing session consists of requests sent to the platform **1040**. As for the exceptions noted above: first, certain links may not be rewritten ("forwarded links"), for instance those which point to resources provided by others than the Merchant and Affiliate; second, it should be noted that URLs for resources which themselves could not include links, e.g., an image file, need not be rewritten to maintain the above-described functionality. Thus, from the composite site browsing state **3300**, the user's browsing session returns to the composite site browsing state **3300** when following non-forwarded links **3350** and leaves the composite site browsing state **3300** when following a forwarded link **3400**.

[0055] **Fig. 4** depicts schematic of a composite site page **4000** as it could be parsed and rendered on a user's browser in accordance with an illustrative embodiment. The composite site page **4000** includes a page from a first peer's on-line catalog **4100** and a first banner **4400** from the first peer. The composite site page **4000** also includes a second banner **4300** from a second peer. In some embodiments, the first peer is a Merchant and the second peer an Affiliate. The composite site page **4000** also includes an offer-related document **4200** that may be created in connection creation of the offer and a sales logo **4500** indicating, for instance, a special sale or promotion.

[0056] One skilled in the art having the benefit of this disclosure will readily appreciate that possible layout of the composite site page **4000** is, in no way limited, to the specific layout shown in **Fig. 4**. Rather, one skilled in the art will now readily apprehend that many others could be made, and apprehend how to make them. Further, the layout of the composite site page **4000** need not be static and could change from request to request.

[0057] **Fig. 5** depicts a block diagram of a reference rewriting module (such as the reference rewriter **1130**) in accordance with an illustrative embodiment. The reference rewriting module rewrites resource references that point to target resources to references that point to platform resources.

[0058] Initially when the platform **1040** receives a request for platform resource **5100** it is passed to a parsing module **5200** that disassembles the reference and extracts an identifier of a target resource. Next a request for the target resource **5300** is sent and a response **5400** received including the target resource. In illustrative embodiments of the invention, the HTTP protocol is used and resources identified with URLs.

[0059] A URL extraction module **5500** parses the response **5400** to identify URLs that should be rewritten. One skilled in the art will appreciate that there are several levels of generality in which URLs can be detected and rewritten/substituted. As an initial matter, the response **54000** could be an ASCII file, a binary file, or a combination. The contents could be in a particular language, e.g., HTML, Java/ECMA Script, or a structured document such as a MACROMEDIA FLASH file, XML file, or PDF file. Still further, the response **5400** could be an unstructured document. In some embodiments, there are plural URL extraction modules **5500** and each is associated with a particular language and handles documents in that language.

[0060] For structured documents, preferably the structure is known, and, if the size of the document is stored as part of the document, this may be extracted as well and rewritten to accommodate changes to the size of the document on account of rewriting URLs. For unstructured documents, parsing may not be possible and pattern matching may be used.

[0061] The URL extraction module **5500** need not extract each URL for rewriting and in some embodiments, URLs that identify documents which themselves do not contain any selectable links, e.g., images, sound files, video clips, are left unaltered.

[0062] For URLs that should be replaced, a set of replacement rules **5600** define how the replacement should be affected. The replacement rules **5600** should be suitably chosen to effectively replace the URLs without introducing

errors into the expected behavior of the user's client system when presenting the altered document. For documents in a particular language, the replacement could be language specific. In HTML, for instance, simply placing the original URL in the data portion of the path of a new URL that references a resource on the platform 1040 could be sufficient. In JavaScript, where the URL could be generated at the client-side after rendering, the replacement rule may rewrite functions generating such URLs to be wrapped inside another function that calls the original function, receives the result, and handles the result as described above for HTML.

[0063] For structured or semi-structured documents, URLs may be substituted as described above for an HTML document, however care should be taken to maintain any internal consistency in the document. For instance, a size field should typically be increased to account for the increased length of the substituted URLs. As noted above, for unstructured documents, search/replace with pattern matching could be used.

[0064] Based on the replacement rules 5600 a replacement reference is determined and a URL replacement module 5700 replaces the original reference with the replacement reference. One of skill in the art will appreciate that, with decreased generality, other cases of reference replacement can be handled. For instance, client-side state objects (commonly known as "cookies") can be replaced so that an original cookie set by a target site is wrapped in a cookie set by the platform. On subsequent requests to the platform, this cookie will be sent, the target cookie extracted and forwarded along with the request for a target resource.

[0065] Another example is secure sockets layer (SSL) sessions. For SSL sessions, the platform initiates an SSL connection with the user's client system as a server and another SSL with the target system as a client; i.e. the platform acts a 'man-in-the-middle' observer. The platform then receives encrypted requests from the client system, decrypts the requests, and submits a suitably reencrypted request to the target system. The response from the target system is decrypted, references are replaced as described above, and then reencrypted and returned as a response to the client. As one of skill in the art will appreciate, the platform should obtain suitable certificates for carrying on in this manner.

[0066] Although the present invention has been described in terms of features illustrative embodiments, one skilled in the art will understand that various modifications and alterations may be made without departing from the scope of the invention. Accordingly, the scope of the invention is not to be limited to the particular embodiments discussed herein, but should be defined only by the allowed claims and equivalents thereof.

ANNEX A

REFERRED TO IN THE APPLICATION

ENTITLED

"A METHOD AND SYSTEM FOR MANAGING NETWORK-BASED PARTNER RELATIONSHIPS"

METHOD AND SYSTEM FOR COMPOSITE SITE RESOURCE GENERATION

FIELD

Features of the invention relate generally to systems for server-side generation of resources in client-server computing and, more particularly, system architectures for generating composite web site resources.

BACKGROUND

With the recent proliferation of electronic commerce systems, a need has arisen for systems providing a convenient and efficient means for automation and management of business relationships between electronic commerce trading partners.

One desirable functionality, particularly in the context of such systems, is the ability for electronic commerce trading partners to provide prospective customers a set of on-line resources that reflects the particular relationship between the electronic commerce trading partners. For instance, if a merchant with an on-line catalog, partners with several affiliates, the merchant may desire to provide collections of on-line resources to prospective customers that are unique for each affiliate. This will frequently also be desirable for the affiliates, as well. Given its current ubiquity, a world wide web site is a conventional and important platform for providing such collections.

Frequently it is desirable to provide collections of resources that include resources from the web trading partners. Frequently, the web trading partners have invested considerable time and money in developing their own site content and having to recreate content in the context of a partner relationship is inefficient and costly. Keeping with the above example, the merchant with its on-line catalog at its website and the affiliate with its own content, desire to present a collection of resources which are a composite of the merchant's catalog pages and the affiliate's content (possibly with other resources as well). Thus, a solution is needed that provides a means to integrate resources from distinct, arbitrarily located and/or arbitrarily chosen, resource collections to provide a composite resource. It is further desirable for such a means to be

amenable to rapid setup to facilitate deployment of collections of composite resources unique to a particular web trading partnership.

5 With some conventional solutions, set-up of these unique collections can be exceptionally burdensome. In a worst case, in order to provide a composite collection of resources, the collection would have to be specifically-created in a one-off manner. Some conventional
10 solutions do exist which can achieve greater efficiencies than this, however, conventional solutions are still lacking.

15 One type of related conventional system involves server-side logic that executes to dynamically create resources. Sun Microsystems' JAVA Server Pages and servlets is one example, as is the PHP environment, as are older CGI-based scripts or program. Such systems typically involve the server-side execution of code for the generation of all or a portion of the
20 resource. In such environments, the resource is local to server system, although it may be dynamically generated. An additional feature that can be incorporated with such systems is database connectivity. This allows for results of database queries to be incorporated within
25 resources. These conventional systems do not provide an effective means for incorporating (possibly dynamic) resources maintained on remote hosts, e.g., the merchant and the affiliate, to form a composite resources. Even with database connectivity, such conventional systems still
30 suffer from a database update problem, namely that hosts possibly contributing resources to be used in a composite resource would have to update a central database storing all such possibly-contributing component resources whenever the component resource was altered. Thus there is a
35 need in the art, for a system to exist that provides an effective means to incorporate arbitrarily-chosen and arbitrarily located component resources.

40 Another conventional solution overcomes some of the problems noted above, but creates others. This solution involves having a monolithic software architecture that is responsible both for managing the relationship among the web trading partners and also generating composite
45 resources. Such a system, however, has performance inefficiencies. First, the portion of the application which generates composite resources can impair performance of the system overall. Second, with a monolithic architecture, distribution of the application becomes hampered. It
50 may be desirable, particularly in high-transaction-volume environments, to distribute portions of

the application across several operating environments. For instance, a first portion responsible
5 for tracking user interaction and generating event data could be implemented under a different
operating system than a second portion responsible for generation of composite resources,
depending on the relative strengths of the operating system, e.g., threading package, memory
10 management, security model, etc.

Accordingly, there is a need for methods and systems that conveniently and efficiently
generate composite resources including arbitrarily chosen/ arbitrarily located component
15 resources, and further assemble the component resources in accordance with a layout particular
to a partner relationship. Still further there is a need for such a means to be decoupled from other
application logic to facilitate distributed processing and application modularity.

SUMMARY

Particular and preferred aspects of the invention are set out in the accompanying
25 independent and dependent claims. Features of the dependent claims may be combined with those
of the independent claims as appropriate and in combinations other than those explicitly set out in
the claims.

The present invention provides a solution to these and other problems with a method and
system for generation of composite site resources. One aspect of the present invention provides
35 computer-controlled methods for generating a composite site resource. An illustrative method
includes receiving an identifier of a composite site and retrieving a characterization of the
composite site resource responsive to the identifier. The characterization includes a set
40 identifiers for a set of component resources. The method also includes communicating requests
to retrieve the set of component resources based on the set of identifiers and receiving the set of
component resources. Then, the component resources are assembled in accordance with the
characterization for creating the composite site resource. The composite site resource is returned
45 to a requesting system.

In a variation on this method, assembling the resource includes assembling a frameset.
50 The component resources include identifiers of resources of frames of the frameset. This

variation also includes receiving requests for the resources of frames of the frameset, retrieving
5 the resources of frames of the frameset; and returning the resources of frames of the frameset.

In another feature, retrieving a characterization of the composite site resource includes
submitting a query to a relational database for the characterization; and transforming results from
10 the query from a relational data model to an XML schema.

The characterization of the composite site resource can be static or dynamic. In a further
15 feature, the characterization of the composite site resource is associated with a relationship
between a first peer entity and a second peer entity. The characterization of the composite site
resource is predetermined by one of the first or second peer entities. In this variation, providing
20 requests for the component resources includes providing a first request for a first component
resource to a first sever system associated with the first peer entity, providing a second request
for a second component resource to a second sever system associated with the second peer entity.

Yet another illustrative method includes receiving a request for a composite site resource,
25 the request comprising an identifier of a composite site, and requesting a characterization the
composite site associated with the identifier. Next, in this method, identifiers of component
30 resources of the composite site resource are received and, the component resources requested
based upon the identifiers. This method then includes receiving and returning the component
resources, receiving the composite site resource; and providing the composite site resource.

Yet another illustrative method includes receiving a request for a resource from a first
35 client system. The request is received with a first server application and includes an identifier of
a composite site. The request is for a composite site resource. The first server system extracts
40 the identifier of the composite site. A message including the identifier of the composite site is
communicated to a second server application for requesting the composite site resource. The
second server application retrieves a characterization of the composite site resource based on the
45 identifier of the composite site. Component resources from the characterization of the composite
site resource are identified and provided to a second client a system. The second client requests
the component resources and returns the component resources to the second server application.
50 The second server application assembles the component resources in accordance with the

characterization of the composite site for creating the resource associated with the composite
5 site. The composite site resource is returned to the first server application; and the first server
application returns the composite resource to the first the first client system.

BRIEF DESCRIPTION OF THE DRAWINGS

10 The above and other features, aspects, and advantages of the present invention will
become better understood with reference to the following description and accompanying
15 drawings of illustrative embodiments, and appended claims, where:

Fig. 1 depicts a flow diagram showing high-level process flow in a composite site resource
generation system with tabular layout;

20 Fig. 2 depicts a flow diagram showing high-level process flow in a composite site resource
generation system with frameset layout;

25 Fig. 3-1 depicts an HTML table-based composite site definition;

Fig. 3-2 depicts an HTML frame-based composite site definition;

30 Fig. 4 depicts a composite site schema;

Fig. 5 depicts a flow diagram of the generation of an HTML-table based composite site
showing schema references;

35 Fig. 6 depicts a flow diagram of the generation of an HTML-frame based composite site
showing schema references; and

40 Fig. 7 depicts a process separation in connection with composite site generation.

DETAILED DESCRIPTION

OPERATING ENVIRONMENT

45 In some embodiments, features of the present invention operate in an architecture for
50 administering and managing network-based partner relationship. Such a system is described in

the concurrently-filed application entitled "Method and System for Managing Network-based
Partner Relationships," attached hereto as ANNEX A. In such an architecture, the present
invention provides an improved system and method for generating a collection of composite
resources (a "Composite Site"). In this context, an end-user interacts with a Composite Site
reflecting a particular network-based partner relationship. In an illustrative embodiment, a
transactional engine is used for tracking the browsing session of a user to monitor activity in the
context of the partner relationship. The resources provided to the client system of the user in the
browsing session are composite site resources generated in accordance with the present
invention. In another aspect of this embodiment, a characterization of the composite site
resources is generated in the connection with establishing the network-based partner relationship.

DESCRIPTION OF FIGURES

Fig. 1 depicts a high-level flow diagram of an illustrative embodiment of a composite site
resource generation system for generation of a composite site with an HTML table layout.
Process flow initiates where a user operating a client system "internet surfer 100" provides a
request 1 comprising an identifier of a composite site ("csid"). The internet surfer 100 operates
client software executing on computing machinery to initiate the request 1 as is known in the art.
No particular hardware/software combination is fundamental. Rather, any suitable computing
platform, including, for instance, a mobile device, set-top box, internet appliance, or general
purpose computer could be used.

A transactional engine 200 receives the request 1. The transactional engine 200
preferably is a collection of software modules executing on a general purpose computer. The
transactional engine 200 performs functions including tracking the browsing session of the
internet surfer 100 in connection with a system for managing network-based business
relationships. ANNEX A of the present disclosure is a concurrently filed patent application
illustrates such a system. One skilled in the art will appreciate that features of the present
invention are not limited to co-operation with features in system in the above-mentioned patent
application. The opposite is true.

The request 1 includes an identifier of a composite site. The transactional engine 200

analyses 2 the request 1. This involves parsing the request 1 and extracting the identifier of the composite site. In some embodiments, the request is an HTTP Request Message, and the
5 identifier of the composite site is passed in the path portion of a URL. Other techniques available to one skilled in the art could also be used. Next, the transactional engine 200 invokes a method 3 in a composite site manager 300 (sometimes abbreviated CSManager) to retrieve a
10 composite site definition associated with the composite site identifier.

The composite site manager 300 comprises software modules for generating composite
15 site resources. Of note is that, architecturally, the CSManager 300 is completely decoupled from the transactional engine 200. This decoupling allows optimization of the transactional tracking features of the transactional engine 200 and the composite site generation features of the
20 CSManager 300. In some embodiments the transactional engine 200 is implemented in a language such as C for relatively high efficiency and the CSManager 300 is implemented in JAVA or other higher level language with increased internetworking functionality. Still further, in
25 embodiments which operate in the context of managing network partner relationships, decoupling of the CSManager 300 from the transactional engine 200 allows those aspects of the logic of the relationship reflected in the composite site to be separated from the transactional
30 engine 200. This increases scalability of both the CSManager 300 and the transactional engine 200 and facilitates their deployment in a distributed environment. For instance, the CSManager 300 could, in this way, be deployed on a machine remotely disposed from the machine executing the transactional engine 200 (with communication across a network).

35 The CSManager 200 retrieves 4 a definition of the composite site from storage (not shown). The composite site definition is described in greater detail below in connection with
40 Fig. 3-1 and Fig. 3-2. Briefly here, the composite site definition comprises identifiers of component resources and other structural information for the composite site resource. In some embodiments, presentation information could also be included. The CSManager 300 extracts the
45 identifiers and provides a component request 5 to the transactional engine 200. The transactional engine retrieves the component resources and responds 6 to the component request 5 with the component resource. In some embodiments, the transactional engine 200 performs
50 transformative processing on the component resource before responding. The component

request-response cycle may be repeated depending on the number of component resources
needed to build the composite site resource.

When the CSManager 300 has received the needed component resources, the CSManager
300 builds 7 the composite site resource and returns 8 the composite site resource to the
transactional engine 200. The transactional engine 200 then provides a response 9 to the request
1 of the internet surfer 100. In some embodiments, the transactional engine 200 performs
transformative processing on composite site resource before providing the response 9 to the
internet surfer 100.

Fig. 2 depicts a high level flow diagram of an illustrative embodiment of a composite site
generation system for generation of a composite site resource with an HTML frame layout. The
process flow differs from that described above in connection with Fig. 1 in that the client system
of the internet surfer 100 first receives a frameset and thereafter the component resources
associated with each of the frames in the frameset.

Process flow up through where the CSManager 300 retrieves 4 the definition of the
composite site is similar to that discussed in connection with Fig. 1. However, in this instance
the composite site definition includes a frameset as is further described below in connection with
Fig. 3-2. The composite site definition includes identifiers of component resources. However in
this instance, the component resources may themselves be references to other component
resources. For instance, in some embodiments a frameset component resource includes a URL to
a document. The CSManager 300 initiates a request 15 for the component resource from the
composite site definition. The transactional engine 200 returns 16 the URL to the component
resource say, a document, as well as the document itself. In some embodiments the URL is
rewritten to facilitate tracking of the user's browsing session as is described in greater detail in
the above-mentioned disclosure set forth as ANNEX A. In some embodiments, the CSManager
300 conveniently caches the component resource to speed retrieval. The above-described
request – response cycle is repeated for the component resources identified in the composite site
definition. When this completes, the CSManager 300 builds the composite site frameset 17 by
appropriately incorporating the URLs into the composite site definition and provides 18 the
composite site to the transactional engine 200.

The transactional engine 200 returns 19 the composite site resource, in this instance the frameset, to the internet surfer 100. The client system of the internet surfer 100 proceeds conventionally to submit request 20 to the locations of the URLs in the frame set for the associated component resources. The transactional engine 200 provides a request 21 to the CSManager 300 for the component resource; the CSManager 300 builds 22 the component resource, preferably retrieving it from a cache, and returns 23 the component resource to the transactional engine 300. The transactional engine 200 then provides 24 the component resource to the internet surfer 100.

In accordance with an illustrative embodiment used in connection with managing a network-based business relationship, the characterization of composite site resource is associated with a relationship between a first peer entity and a second peer entity. The characterization of the composite site resource is predetermined by either the first or second peer entities, or they both could participate in establishing the characterization. Component resources could be requested from server systems operated by both the first and second peer entities when creating the composite site resource.

Fig. 3-1 depicts an HTML table-based composite site definition 3000 in accordance with an illustrative embodiment. The HTML table-based composite site definition 3000 includes a first component resource identifier 3100 and a second component resource identifier 3200. The particular syntax or number of component resource identifiers is not fundamental and may vary. Layout techniques are not limited to HTML tables or framesets. In other embodiments, layout elements in cascading style sheets or style languages could be used. More generally, layout techniques available to one skilled in the art and able to be parsed and rendered by the client system of the internet surfer 100 could be used.

Fig. 3-2 depicts an HTML frame-based composite site definition 3500 in accordance with an illustrative embodiment. The HTML frame-based composite site definition 3500 includes a third component resource identifier 3250, a fourth component resource identifier 3300, and a fifth component resource identifier 3400 (as well as others not specifically referenced). The particular syntax or number of component resource identifiers is not fundamental and may vary. As noted above, in the case of the HTML frame-based composite site definition 3500,

component resource identifiers could be replaced by the CSManger 300 with URLs and the browser application of the internet surfer 100 subsequently request the resources associated with the URLs.

Fig. 4 depicts a composite site schema 4000 in accordance with an illustrative embodiment. A 'CSManager' class 4100 is the main class of the CSManger 300. The 'CSManager' class 4100 initializes the CSManger and creates a thread for a server manager handling requests from the transactional engine 200. A 'CSBuilder' class 4100 performs the function of building the composite site and is decoupled from the communication between the CSManger 300 and the transactional engine 200. A 'CSFactory' class 4150 creates a new instance of a 'CompositeSite' class 4200 when needed.

The 'CompositeSite' class 4200 represents the composite site definition. In some embodiments, the presentation of the composite site is in HTML, in others it could be in XML and have an accompanying style sheet, in still others it could be a markup language for use with wireless devices, and still other formats could be used.

A composite site resource can have more than one representation; there is no representation definition in the 'CompositeSite' class 4200. A 'Page' class 4300 defines the representation of the composite site. There is a link to the 'Page' class 4300 from the 'CompositeSite' class 4200. There, could be different types of the 'Page' class 4300. In some embodiments, there is only an 'HTMLPage' subclass 4350. The 'HTMLPage' subclass 4350 is a specialization of the 'Page' class 4300 and represents the page associated with a composite site in an HTML format. The build method of this subclass is responsible for building the composite site with an HTML representation.

An abstract 'Layout' class 4400 provides an association between the different component resources of the composite site and their situation in a page. A 'FormatConstraint' class 4450 is associated with the 'Layout' class 4400. The 'FormatConstraint' class 4450 represents constraints for a format associated with a Layout. Format constraints for an HTML representation may include, for instance, a FrameBorder, a Scrolling property, a Background Color, a Resizability property, and a Splitability property. To one Layout different

'FormatConstraint' classes 4450 could be associated depending on the desired format.

5 In some embodiments, an 'IComponent' class (not shown) represents common attributes and/or behaviors common to plural component resources. Subclasses having a component resource as their content can then inherit and extend this class. The content of a component
10 resources may have plural representations. The representations may be context dependent and/or format dependent. Preferably, only a single external reference is used to access a component resource, irrespective of representation.

15 In some embodiments, common attributes include a component resource type, a component resource content, and a component resource external reference. An illustrative embodiment has four primary component resource types: an HTML document, an image, a
20 clickable image—namely a hypertext link for which an image is the anchor—and text.

A 'Component' class 4500 is a super class of the component resources. The
25 'Component' class 4500 includes an association to a 'Locator' object 4550. For a static component resource, the location could correspond to the URL; for a dynamic component resource the location of the component could be variable. To the 'Locator' object 4550, an
30 extraction rule could be associated. This rule determines the location of the component as function of a set of parameters.

35 A 'Container' class 4600 contains a set of component resources and has an associated 'Grouping Rule' object 4625 and 'Selection Rule' object 4650. These are for the suggested functions, grouping and selecting among the component resources, respectively.

40 Fig. 5 depicts a flow diagram of the generation of an HTML-table based composite site showing schema references in accordance with an illustrative embodiment. Process flow initiates in a ServerManager thread 5100 that invokes a getCompositeSite method 51 in a
45 CSManager object 5200. The CSManager object 5200 invokes a getCompositeSite method 52 in a CFactory object 5300 that takes as its argument an identifier of the composite site for generation. A new Composite Site object 5400 is returned 53 and its build method 54 invoked. Next, a build method 55 is invoked in a Page object 5500 which, in turn, invokes a getData
50 method 56 in an associated component object 5600.

Assuming the component resource is a document with associated URL, a `getComponent` message 57 is passed to a client manager 5800 in the transactional engine 200 that fetches and returns the fetched document 59. An `analyzeDocument` method 59 executes for parsing the document and the `getData` method 56 completes with the document data returned 60 to the Page object 5500.

The 'Page' object 5500 invokes a `getConstraint` method 61 in a Layout object 5700 in completing execution of the build method 55. When execution completes the page is returned 62 to the Composite Site instance 5400 that returns the results 63 to the CSManger 5200 that in turn provides the results to the ServerManager thread 5100.

Fig. 6 depicts a flow diagram of the generation of an HTML-frame based composite site showing schema references in accordance with an illustrative embodiment. Process flow initiates in a ServerManager 6050 that receives a message from the transactional engine 200 for building the composite site. A 'getCompositeSite' message 71 is sent to a CSManger object 6100 that invokes a `getCompositeSite` method 72 in a CSFactory object 6150 and provides an identifier of the composite site for creation. (Note that, function differences are not implied by different a reference numeral for the CSManger from that in Fig. 1 only clarification when reference is being made in the context of the schema). The CSFactory object 6150 returns 73 a new Composite Site instance 6200 and the CSManger object 6100 invokes a build method 74 in the new Composite Site instance 6200. A build method 75 is invoked in a page object 6300 that, in turn, invokes a `getURL` method 76 in a component object 6500 to retrieve the URL (component resource) associated with the page being assembled. The URL is passed along with a `parseURL` message 77 to a client manager 6600 in the transactional engine 200 that returns 78 the parsed URL 78 and execution of the `getURL` method 76 completes. The page object 6300 retrieves layout constraints 80 from a layout object 6400 and the builds the composite site page 81 accordingly.

Execution of build method 75 the CompositeSite object 6200 invoked completes and the composite site page is returned 82 and is routed back (steps 83 and 84) to the browser of the internet surfer 100.

As Fig. 6 depicts the situation of a frameset, the browser application conventionally submits requests for the resources associated with each frame of the frameset from the transactional engine 200. The transactional engine 200 provides messages to the CSManager object 6100 to request the component resources. A GetComponent method 85 taking an identifier of the composite site, an identifier of the component resource, and a type of the component resource is invoked in the CSManager object 6100. The CSManager object 6100, in turn, invokes a GetComponent method 86 in the new Composite Site instance 6200.

Next, a build method 87 in the page object 6300 invokes a getData method 88 in the component object 6500 to request the component resource data. Assuming for illustration, the component resource is an HTML document, a GetComponent message 89 is sent to the client manager 6600 of the transactional engine 200 to retrieve the component resource associated with the particular URL. The transactional engine 200 returns 90 the HTML document component resource and the getData method 88 completes. The page object 6300 parses the component resource 92 retrieves any layout constraints 93 and assembles the composite site page 94. The page is returned. The build method 87 completes and the component resource continues back to the browser of the internet surfer 100 (in steps 95, 96, and 97).

Fig. 7 depicts a process separation in connection with composite site generation. As noted above, yet another characteristic feature of the present invention lies is the fact that creation of composite site resources can be decoupled from applications and/or services which use the composite that resources. These aspects are illustrated in Fig. 7 with reference to an illustrative embodiment in which the transactional engine 200 uses the CSManager 300 for the creation of composite site resources.

Process flow initiates when a transactional engine management module 7200 receives a request to provide a composite site resource. The management module 7200 provides a message 7010 to a document construction module 7300. The document construction module 7300 sends a message 7020 to a communication client 7700 to get the composite site resource.

The communication client 7700 communicates the request to get the composite site resource to the CSManager 300 through a sockets layer. The CSManager 300 executes as has

5 been previously described and submits a request 7030 for component resource 7040 of the composite site. The communication client 7700 provides the request for the component resource 7040 to a mkdocs module that instructs a client application 7500 to get the requested resource 7060. The client application 7500 returns 7070 the requested resource.

10 If the document should be parsed, a parsing manager 7600 may receive 7080 the document and return a structure 7090 containing the parsed document. The component resource returns 7100 to the communication client 7700 that provides 7110 the component to the
15 CSManager 300. The CSManager builds 7120 the composite site resource as has been described above, and returns 7130 the composite site resource to the transactional engine 200.

20 It will be apparent from the foregoing that the CSManager 300 is decoupled from the transactional engine 200. These two software components exchange information but do not depend on each other for their internal operations. When the transactional engine 200 performs
25 functions of monitoring and tracking the browsing session of a user as the user browses composite site resources, decoupling of the CSManager 300 provides substantial benefits. At the simplest level, the CSManager 300 can be implemented on a dedicated system for improved efficiency, security, or scalability. In addition, this decoupling allows the transactional engine
30 200 to be optimized for tracking the browsing session.

35 Although the present invention has been described in terms of features illustrative embodiments, one skilled in the art will understand that various modifications and alterations may be made without departing from the scope of the invention. Accordingly, the scope of the invention is not to be limited to the particular embodiments discussed herein, but should be
40 defined only by the allowed claims and equivalents thereof.

Claims

What is claimed is:

- 5 1. A computer-controlled method of constructing a composite site resource, said computer-controlled method comprising:
- 10 receiving an identifier of a composite site;
- retrieving a characterization of said composite site resource responsive to said identifier, said characterization comprising a set identifiers for a set of component
- 15 resources;
- communicating requests to retrieve said set of component resources based on said set of identifiers;
- 20 receiving said set of component resources;
- assembling said component resources in accordance with said characterization for creating said composite site resource; and
- 25 returning said composite site resource.
- 30
2. The computer-controlled method according to claim 1 wherein assembling said resource comprises assembling a frameset, said component resources comprise
- 35 identifiers of resources of frames of said frameset, and said method further comprises:
- 40 receiving requests said resources of frames of said frameset;
- retrieving said resources of frames of said frameset; and
- 45 returning said resources of frames of said frameset.
- 50 3. The computer-controlled method according to claim 1 or claim 2 wherein retrieving a characterization of said composite site resource comprises:
- submitting a query to a relational database for said characterization; and
- 55

transforming results from said query from a relational data model to an XML
schema.

5

10

4. The computer-controlled method according to any of claims 1 to 3 wherein said
characterization of said composite site resource is static.

15

20

5. The computer-controlled method according to claim 4 wherein said characterization of
said composite site resource is associated with a relationship between a first peer entity
and a second peer entity; said characterization of said composite site resource is
predetermined by one of said first peer entity and said second peer entity, and
providing requests for said component resources comprises:

25

providing a first request for a first component resource to a first sever system
associated with said first peer entity; and

30

providing a second request for a second component resource to a second sever system
associated with said second peer entity.

35

6. A computer-controlled method constructing a composite site resource, said
computer-controlled method comprising:

40

receiving a request for a composite site resource, said request comprising an
identifier of a composite site;

45

requesting a characterization said composite site associated with said identifier;

receiving identifiers of component resources of said composite site resource ;

requesting said component resources based upon said identifiers;

50

55

receiving and returning said component resources;

receiving said composite site resource; and

providing said composite site resource.

7. A computer-controlled method constructing a composite site resource, said computer-controlled method comprising:
- receiving a request for a composite site resource from a first client system, said request received with a first server application, said request comprising an identifier of a composite site;
- extracting said identifier of said composite site from said request, by said first server application;
- communicating a message comprising said identifier of said composite site to a second server application for requesting said composite site comprising said resource;
- retrieving a characterization of said composite site, said characterization retrieved by said second server application responsive to said identifier of said composite site;
- retrieving a characterization of said composite site, said characterization retrieved by said second server application responsive to said identifier of said composite site;
- identifying component resources from said characterization of said composite site, providing to a second client identifiers of said component resources;
- requesting said component resources, by said second client, and returning said component resources to said second server application;

assembling said component resources in accordance with said characterization of
said composite site for creating said composite site resource;

returning said composite site resource to said first server application; and

returning said composite site resource to said first client system.

8. A computer-implemented system for constructing a composite site resource
comprising:
- a first server application, said first server application configured for receiving a request
comprising an identifier of composite site, extracting said identifier, and providing a
request for said composite site resource;
- a composite site generation application, said composite site generation application
configured for receiving said request for said composite site resource from, retrieving
a characterization of said composite site based on said request, and providing requests
for components of said composite site resource;
- a first client application, said first client application configured for receiving said
requests for components, said client application configured for retrieving said
components and providing said components to said composite site generation
application;
- wherein, said composite site generation application assembles said components into
said composite site resources in accordance with said characterization.

9. The system according to claim 8 further wherein:

said composite site generation application comprises:

a composite site object for representing a definition of the composite site;

a page object for defining a representation of the composite site; and
5 a layout object for providing an association between said components.

10

15

20

25

30

35

40

45

50

55

ABSTRACT OF THE DISCLOSURE**METHOD AND SYSTEM FOR COMPOSITE SITE RESOURCE
GENERATION**

10

Disclosed are computer-implemented methods and systems for generating composite resources of a web site. A request including an identifier of a composite site is received and the identifier used to retrieve a characterization of the composite site. The characterization includes identifiers of component resources that make up the composite site resource as well as the layout in which the components should be assembled to form the composite site resource. Upon identifying the components, this information is passed to a client system that retrieves the component resources, e.g., from across a network. The retrieved components are assembled into the composite site resource at the server side, of a client-server interaction with the end-user. The composite site resource is provided to the end user's client system. An additionally disclosed aspect is where the composite site resources includes an HTML frameset. A further disclosed feature is cooperation between an application for managing generation of composite site resources and a transaction engine through which the end-user browses composite site resources, and which monitors end-user browsing session events. As an additional aspect, the decoupling of composite site generation management from the functionality of the transaction engine allows for each component to be optimized and an overall system deployed in a more scalable and easily distributable manner.

40

Fig 1

45

50

55

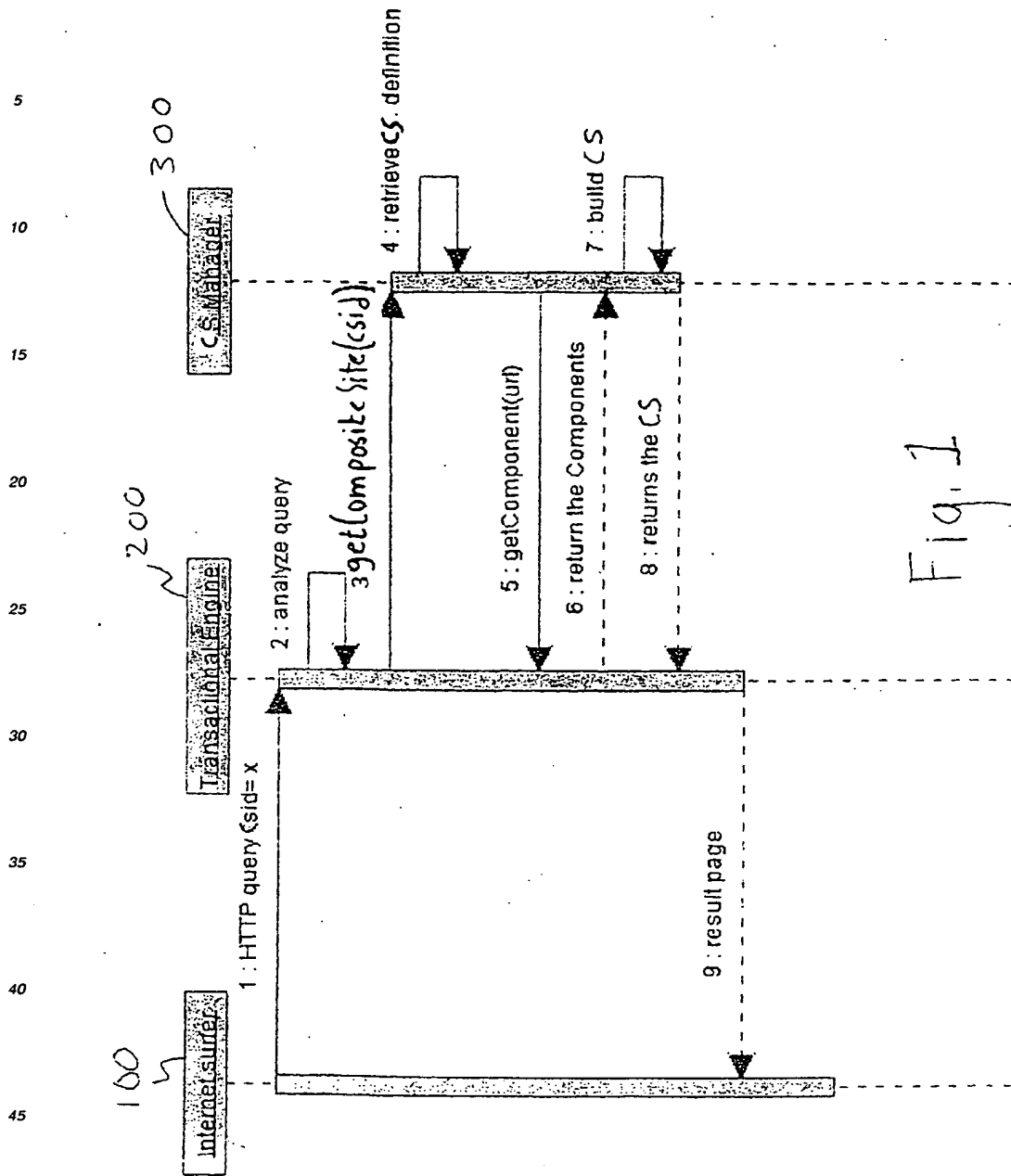
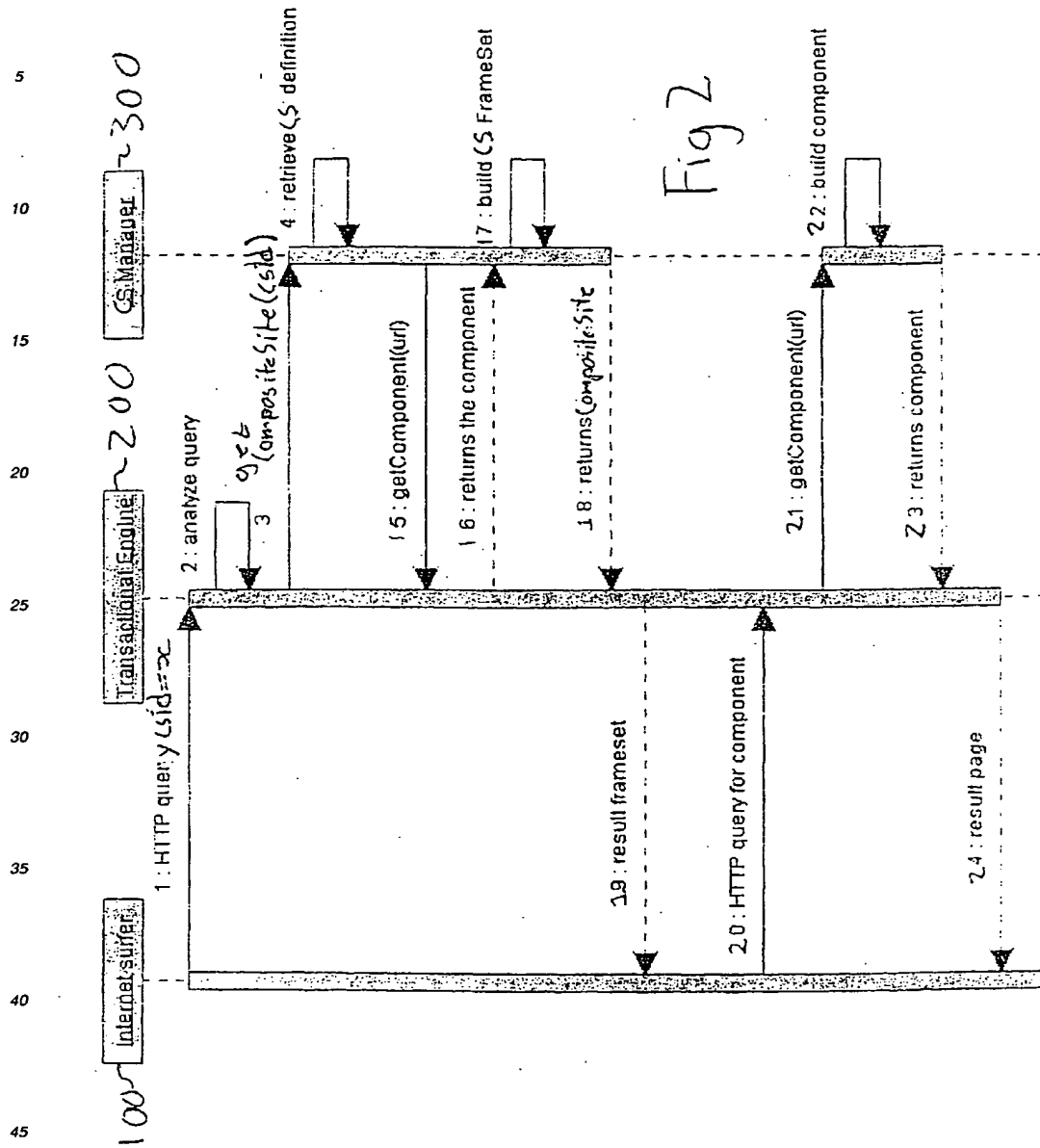


Fig. 1



```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
  <HEAD>
    <TITLE> iChannel Front End </TITLE>
  </HEAD>
  <BODY>
    <TABLE BORDER="1" WIDTH="100%" CELSPACING="0" CELLPADDING="0">
      <TR>
        <TR>
          <TD bgcolor="red">
            ?(Document_1052)
          </TD>
        </TR>
        <TR>
          <TD>
            ?(Document_1)
          </TD>
        </TR>
      </TR>
    </TABLE>
  </BODY>
</HTML>
```

Fig. 3-1

3000

3100

3200


```

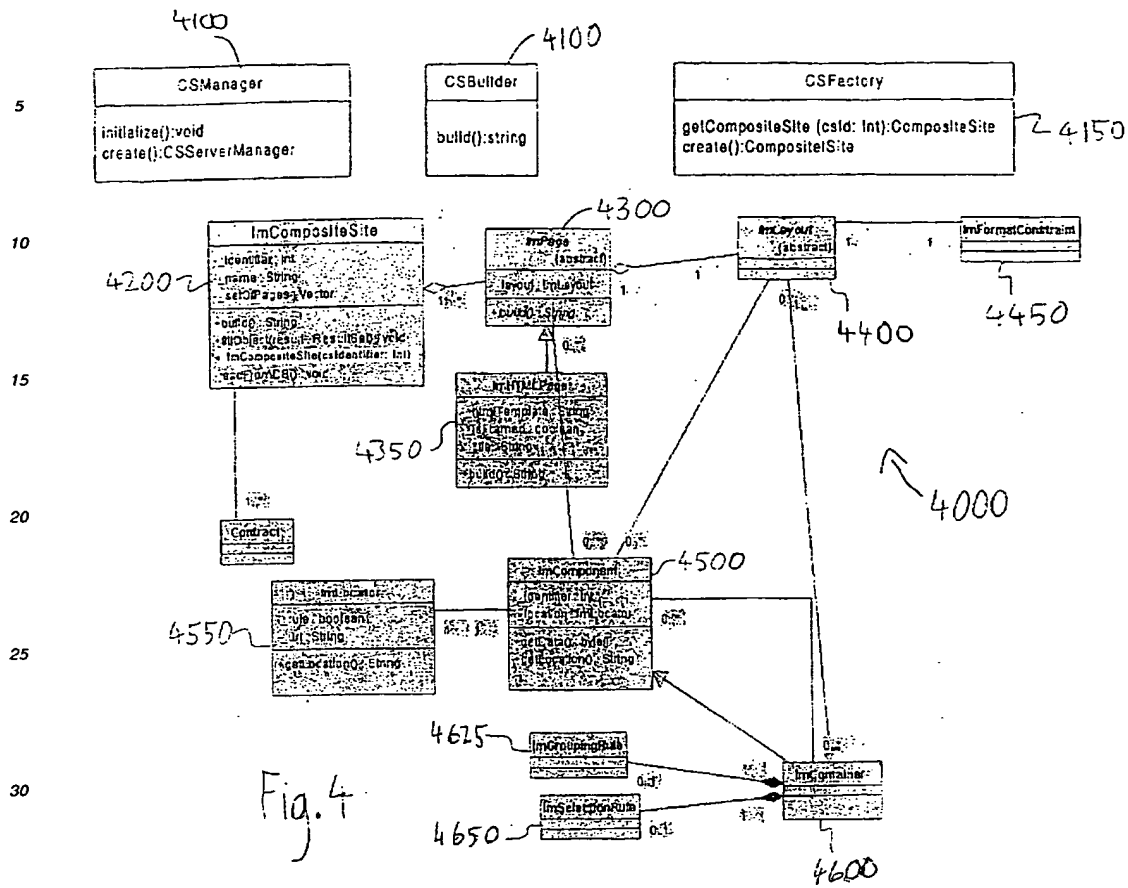
5
10
15
20
25
30
35
40
45
50
55

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>?(DocumentTitleTag_1)</TITLE>
    ?(DocumentHeader_1)
</HEAD>
<FRAMESET rows="10.233393%,*" cols="*">
  <frame name="north" frameborder="1" scrolling="AUTO" src="?(Document_1009)">
  <FRAMESET rows="84.60%,*" cols="*">
    <FRAMESET rows="*" cols="13.408522%,*">
      <frame name="/south/north/west" frameborder="1" scrolling="AUTO" src="?(Document_1014)">
      <FRAMESET rows="*" cols="83.936325%,*">
        <frame name="/south/north/east/west" frameborder="1" scrolling="AUTO"
        src="?(Document_1)">
        <frame name="/south/north/east/east" frameborder="1" scrolling="AUTO"
        src="?(Document_1010)">
      </FRAMESET>
    </FRAMESET>
  <frame name="/south/south" frameborder="1" scrolling="AUTO" src="?(Document_1007)">
  </FRAMESET>
</HTML>

```

Handwritten annotations:

- 3500 (with arrow pointing to Document_1009)
- 3300 (with arrow pointing to Document_1014)
- 3250 (with arrow pointing to Document_1)
- 3400 (with arrow pointing to Document_1010)
- Fig. 3-2



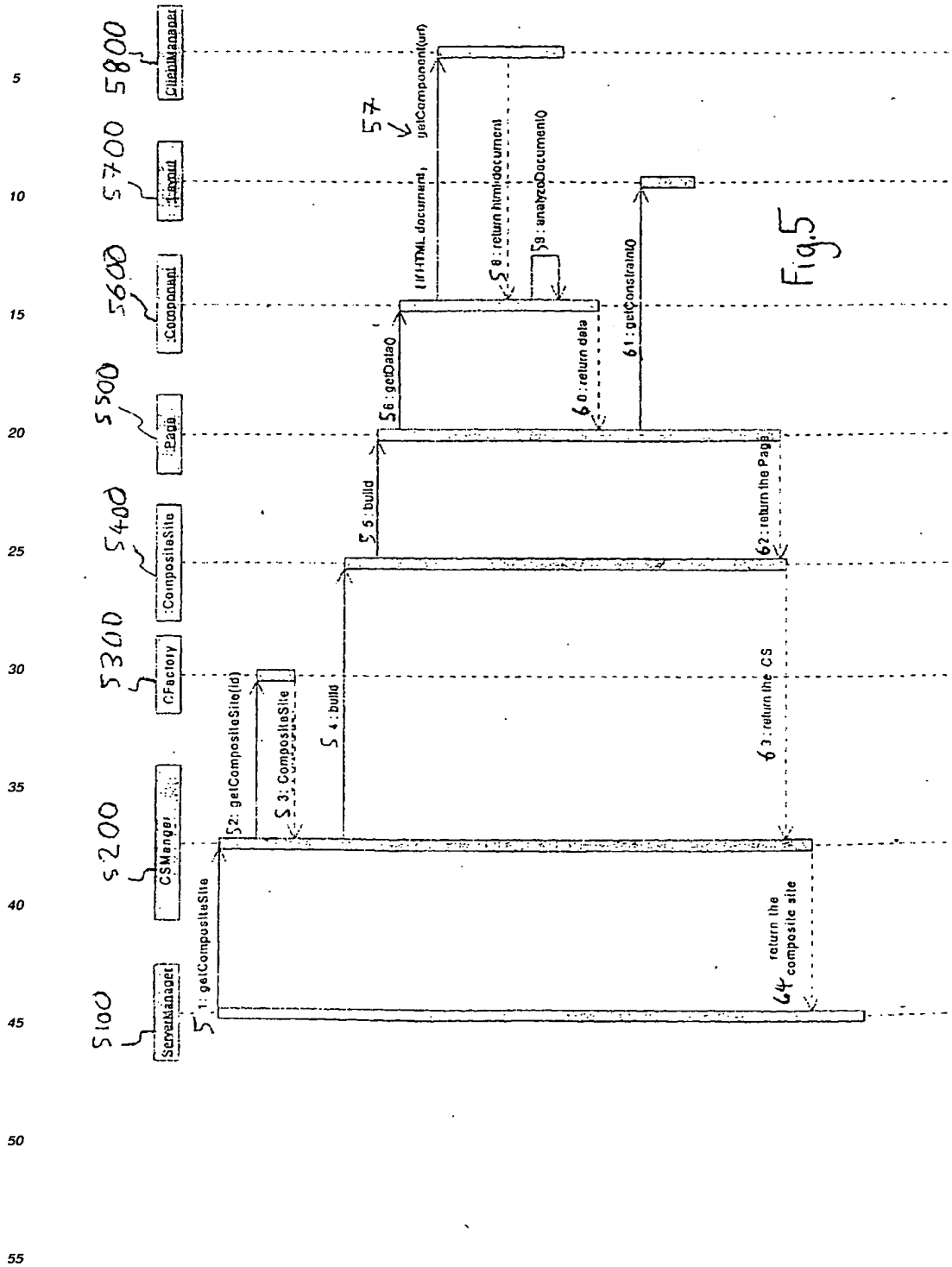
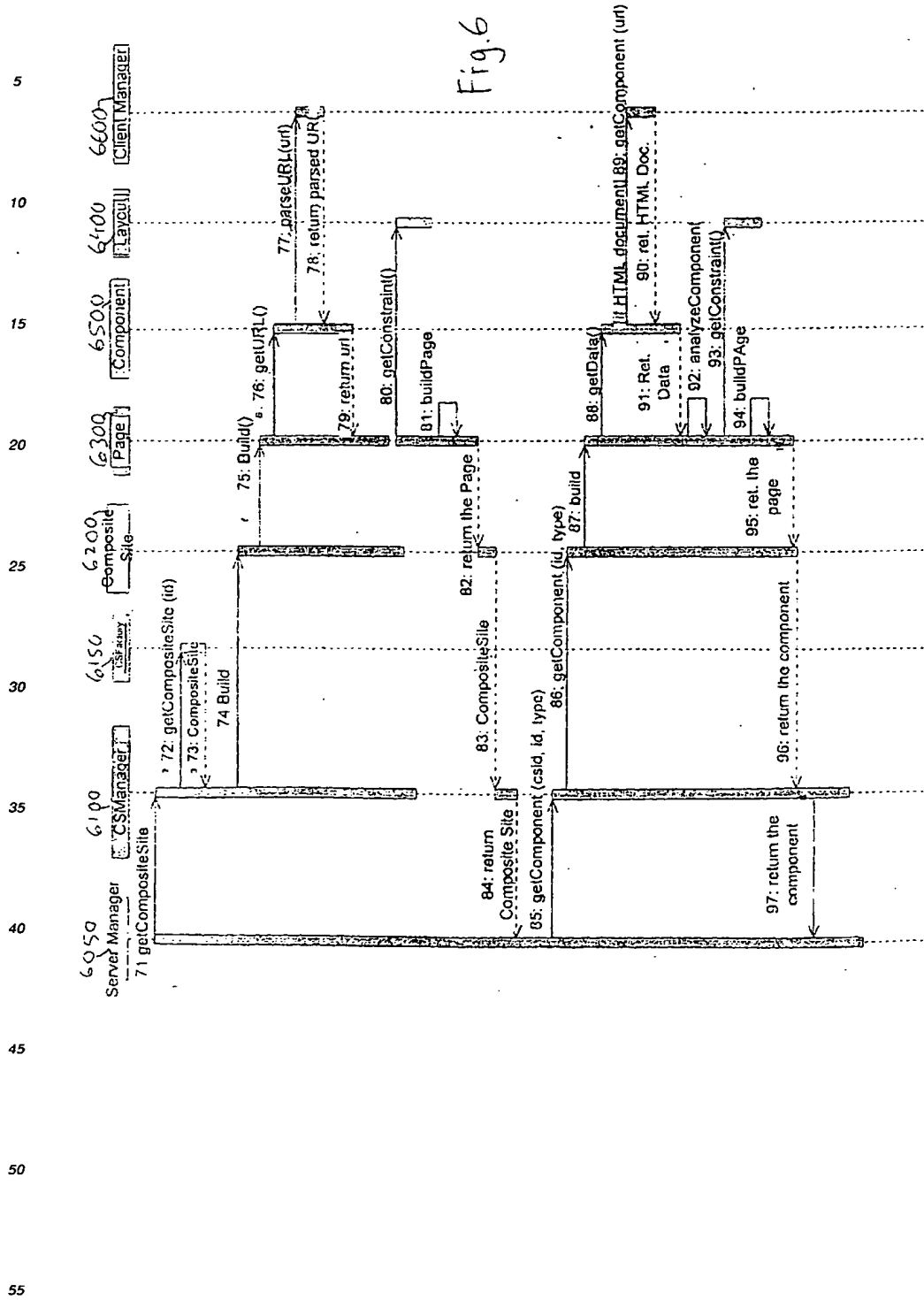
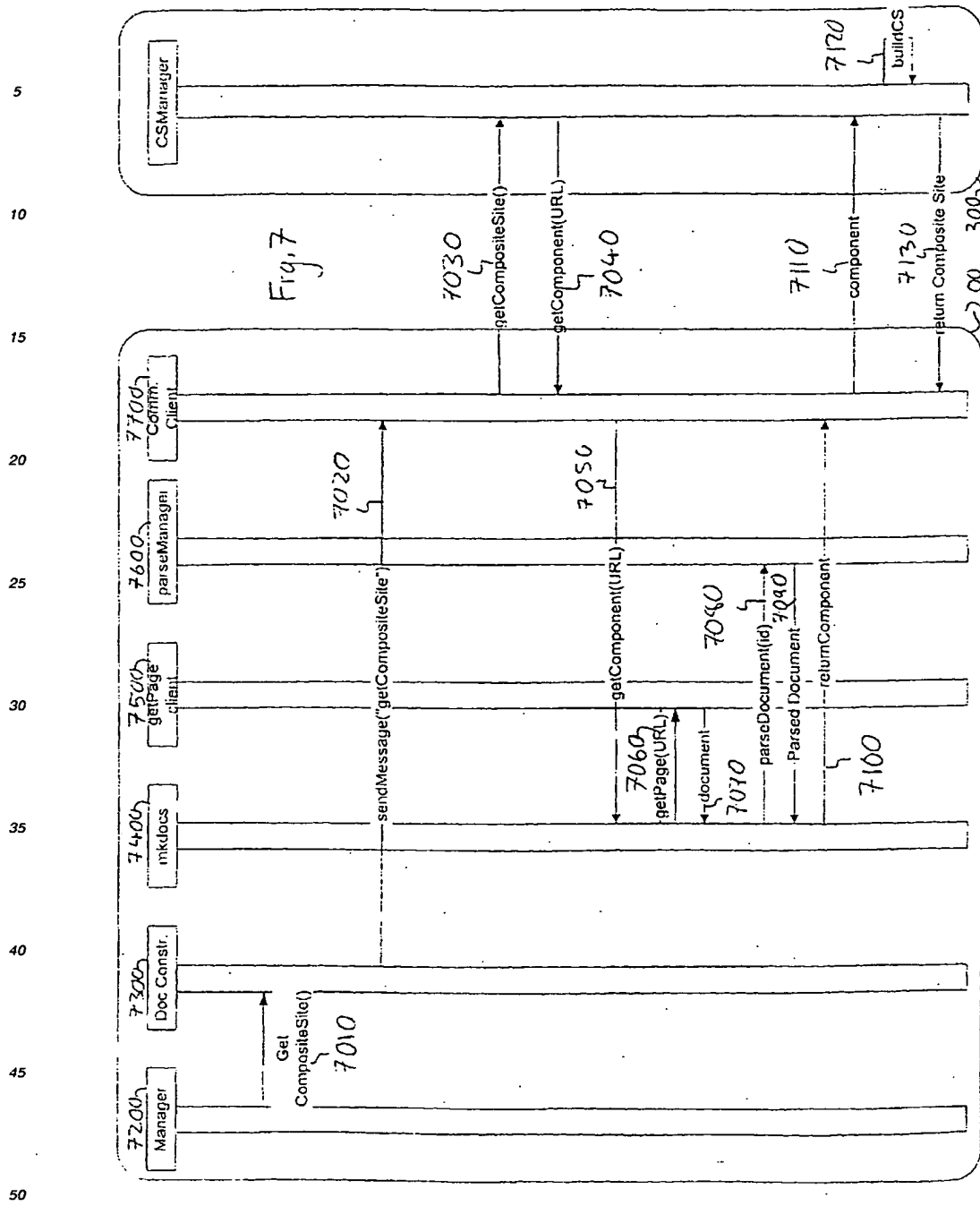


Fig. 6





ANNEX B

5

REFERRED TO IN THE APPLICATION

10

ENTITLED

"A METHOD AND SYSTEM FOR MANAGING
NETWORK-BASED PARTNER RELATIONSHIPS"

15

20

25

30

35

40

45

50

55

METHOD AND SYSTEM FOR TRANSFORMING SESSION DATA

FIELD

Features of the invention related generally to management of browsing session data and more particularly to systems and techniques for the flexible and efficient transformation of session data into an enterprise context.

BACKGROUND

Recent years have seen a trend toward sectors of our economy and society implementing their operations electronically. Electronic commerce has flourished and continues to grow in importance. More generally “virtual” enterprises have also seen rapid growth. Implementing electronic commerce and virtual enterprises has created a need for rapidly-deployable comprehensive solutions for managing partnerships among entities collaborating across computer networks. Features in the co-pending patent application noted above are an example, although the present invention is not limited to cooperation with that system.

Commonly, a user of the services of an electronic enterprise interacts with the enterprise through some client system, e.g. a web browser, mobile phone, set-top box, etc.. On the other (server) side of the interaction are the enterprises’ computing systems. These systems generate data as part of the interaction. This data is driven by the client-server request/response cycle and reflects a low-level aspect of the interaction. More particularly, data from request/response events directly has the limited context of the client-server data flow. Accordingly, for any semantic which spans many request/response cycles, events generated from one request/response cycle represent information in that semantic.

When the electronic enterprise—or a partner of the electronic enterprise—seeks to use information gained in connection with services provided through client-server communication with the user, a fundamental information context problem arises. At the simplest level, event data gained from client-server requests or responses is generally protocol-dependant. More fundamentally, data taken from the client-server data flow is limited to the context or semantic of the client-server data flow. The enterprise, by contrast, typically desires that facts be generated

in other contexts or semantics for use in the enterprise's operations. That is, the enterprise
desires protocol-independent facts that are meaningful in the context of the enterprise's
operations.

For example, a business may desire receiving the fact that a new customer directed to the
business from a particular business partner just bought four items. In this example the fact has
meaning in the context of, e.g., business partnership arrangements, customers, and purchases. It
could be derived from, e.g., an individual navigating with his or her web browser to an on-line
item catalog, selecting links or taking other actions for making purchases, and then ending the
shopping session, say by a conventional checkout function. To further elaborate the point, data
obtained from each HTTP client-server request/response cycle in this browsing session would be
dependant on the HTTP protocol; further, its direct contextual meaning would be similarly
dependant on the context of this request/response cycle. Fundamentally, if events are to be
generated from client requests or server responses, then any semantic that spans more than one
request/response cannot be expressed as such an event. In order, to generate facts meaningful to
the business, aggregations, transformations, or other operations need to be performed based on
events obtained from several HTTP request/response cycles. Referring still to the example
above, facts in the context of the business' arrangement with its business partner, or in the
context of the particular customer need to be gleaned, from a set of HTTP request/response
cycles, each of which individually has the limited context or semantic of that cycle.

Returning from the example, there is a need for a method and system to transform event
data generated in connection with client-server request/response into facts meaningful in
contexts used by the enterprise. Still further, commonly the enterprise itself may be distributed
and/or have partners that are distributed. In order for facts in enterprise contexts to be
beneficially used, there is a further need for such a method and system to be conveniently
implemented in a distributed manner. Further, there is a need for such a method and system to
scale well. When the volume of client-server traffic is high, processing event data for the
generation of enterprise facts could degrade performance of the client-server data flow. It is
further desirable that generation of enterprise facts could be carried on asynchronously from the
client-server data flow to further isolate the processing underlying the generation of enterprise
facts from the client-server data flow. Such a feature could not only improve scalability, but also

distributed implementation.

5 One conventional solution is to write session event data to some persistent storage such
as a database. This solution has deficiencies. Most notably this solution scales poorly. Also,
this solution implies storage of much data that is not of direct interest to the enterprise and
10 further additional processing on this data to correlate this data in a valuable manner. On the one
hand, writing the session-level semantic data to a single shared database introduces unacceptable
database latency into the overall operation of the system. This is only exacerbated if the
15 enterprises' systems are distributed. Further, the sheer volume of session-level semantic data
makes the database solution expensive and unwieldy. On the other hand, if preprocessing is
done and enterprise facts written to the database these deficiencies are not eliminated. When
20 many entities need to access the shared data—particularly when only a small portion may be
relevant to the particular accessing entity—this shared database again becomes a bottleneck. The
database itself could be replicated and propagated, however this would introduce needless cost
and latency as well. Thus there is a need for a scalable means for transforming session-level
25 semantic event data into enterprise facts, and still further, a need for such means to be able to
efficiently distribute enterprise facts to participating entities in an enterprise.

30 Another deficiency with conventional solutions is their lack of modularity and
commensurate difficulty in optimization. One conventional approach to obtaining enterprise
facts from such systems is to incorporate the logic necessary to generate this into the computing
35 systems implementing client-server communications. However, this logic can be relatively
computation and memory intensive. Thus to perform this logic in connection with real-time
operation of the system can result in unacceptable performance decrease.

40 Still further, there are additional difficulties in attempting to rapidly distribute enterprise
facts. In particular some enterprise facts can only be created based on several session-level
semantic events that are greatly disparate in time which makes this attempted solution difficult to
45 achieve without reintroducing one of the deficiencies mentioned above. Thus there is a need for
distributing enterprise facts based on low-level semantic events that are disparate in time without
again encountering the deficiencies in conventional solutions.

SUMMARY

5 Particular and preferred aspects of the invention are set out in the accompanying independent and dependent claims. Features of the dependent claims may be combined with those of the independent claims as appropriate and in combinations other than those explicitly set out in
10 the claims.

These and other problems are solved and benefits obtained by the present system and method transforming session data. In accordance with aspects of the invention, events generated
15 in connection with client-server requests and/or responses including, for instance, HTTP Request and Response messages are transformed into enterprise facts. The term "session-level semantic event 'SLSE'" is used to refer to such protocol-dependent events, generated on the client-server
20 request/response cycle, that cannot express a semantic that spans plural request/response cycles. The term "Enterprise Fact 'EF'" is used to refer to the protocol-independent data, generated upon the satisfaction of predetermined conditions, and expressing an interpretation of a set of SLSEs.

25 One aspect of the invention involves computer-controlled methods of generating enterprise facts from session-level semantic events. An illustrative method includes generating session-level events from data flow of a browsing session of a user. The session-level events are
30 received and accumulated in a storage object. The storage object could be non-persistent. Then, the method determines if one of the session-level semantic events is a triggering condition of one or more rules used to generate enterprise facts. Enterprise facts are then returned for those rules
35 for which the triggering condition is satisfied.

An additional characteristic feature of this illustrative method is where the storage object
40 is associated with the browsing session, or the user. Yet another characteristic feature includes broadcasting enterprise facts across a message broker. Another characteristic feature involves the one or more rules having a condition portion where the condition portion has a first event
45 portion and a second event portion. The first event portion is associated with a triggering event for the rule and the second event portion associated with other events.

Yet another characteristic feature of this illustrative method involves the one or more
50 rules having a first action part and a second action part. The first action part includes a default

55

action indicating an enterprise fact to return and the second action part indicating a function to be performed on second enterprise fact. In another aspect, the second action part may include a function type, a enterprise fact type to be affected; and a field of the enterprise fact to be affected. More generally, a triggered action can access and influence any event accumulated in its container.

Yet another illustrative method involves receiving events created from a user session and storing a plurality of rules, the rules having a condition part and an action part. The condition part is satisfied responsive to selected ones of the events from the user session, and the action part is for creating business event data. Next in this method, the events created from a user session are matched with the particular rules for which the events contribute to satisfaction of the condition part. The method also involves generating business event data through executing the action part of the rules upon determining the condition part of the rules is satisfied and publishing the business event data on a message broker.

Yet another aspect of the invention are computer-implemented systems for generating enterprise facts from session-level semantic events. An illustrative system includes a storage object having a method for storing a session-level semantic event, and a method for testing if the session-level semantic event is stored. The storage objects could be non-persistent. Also included is a enterprise fact generation object having a method for invoking the method testing if the session-level semantic event is stored in the storage objects to determine if a condition of a predetermined rule is satisfied. This system further includes a rule object comprising a set of conditions, a trigger condition for activating a rule; and an action for execution upon activation of the rule.

In another characteristic feature of the illustrative system the enterprise fact generation object includes a data structure for storing a value associated with a key; the key is the trigger condition, and the value is the rule.

Yet another characteristic feature includes a first message broker, where the message broker is configured for receiving enterprise facts from the enterprise fact object.

Another characteristic feature of this system involves the one or more rules having a first

action part. The action part includes a default action indicating a first type of enterprise fact to return and a second action part indicating a function to be performed on second type of enterprise fact. In another aspect, the second action part may include a function type, a enterprise fact type to be affected, and a field of the enterprise fact to be affected.

Another aspect of the invention is a computer programmed to carry on functions as described above in connection with the illustrative system and methods. Programmed general purpose computers may generally provide structures for performing functions in accordance with features of the invention. An illustrative computing apparatus includes means for receiving event data from a user's session; means for accumulating the event data; means for determining if one of the session-level semantic event datum is a triggering condition of one or more rules for generating high-level semantic; and means for returning a enterprise fact for those of the one or more rules for which the triggering condition is satisfied.

BRIEF DESCRIPTION OF THE DRAWINGS

These and other aspects, features, modes, and benefits will be better understood with reference to the accompany detailed description of illustrative embodiments and figures where:

Fig. 1 depicts a block diagram of an operating architecture for efficient generation of enterprise facts ("EF"s) from browsing session event data;

Fig. 2 depicts architectural components in Fig. 1 in greater detail;

Fig. 3 depicts a flow diagram of an illustrative operational scenario for efficient generation of EF from SLSE;

Fig. 4 depicts a block diagram of an operating architecture for efficient generation of EFs from browsing session event data in accordance with another embodiment; and

Fig. 5 depicts a flow diagram of an illustrative operational scenario for efficient generation of EF from SLSE in accordance with embodiments illustrated by Fig. 4.

DETAILED DESCRIPTION

5 With reference to the figures, Fig. 1 depicts a block diagram of an operating architecture for efficient generation of enterprise facts from browsing session event data in accordance with an illustrative embodiment. A first message broker 1100 provides session level semantic events 1300 (hereinafter "SLSE") from across a network 1200. In some embodiments, the message
10 broker 1100 uses a publish/subscribe model, in others a queue based model, and still others could be used. In some embodiments Java Messaging Services are used, while this is not fundamental, and other messaging APIs could be used. The presence of the network 1200 is not fundamental, and in some embodiments the SLSE 1300 are retrieved directly from a memory shared with the source of the SLSE 1300.
15

20 In some embodiments the SLSE 1300 include a data structure that comprise an event identifier, an event type, and event attributes. Some events are canonical with fixed, well-known attributes. No particular representation of the SLSE 1300 is fundamental; a representation in the Extensible Markup Language ("XML") could be used as could an object representation, or other
25 representations available to one skilled in the field. Still further, there is no fundamental limitation on the number of types of the SLSE 1300 and new types could be created.

30 A set of storage objects 1400 receives the SLSE 1300. The SLSE 1300 are filtered into members of the set of storage objects 1400 based on a policy. In some embodiments, the policy is session-based, namely there is one storage object in the set of storage objects 1400 for each
35 browsing session. New storage objects 1400 are created as necessary as the number of sessions increase and destroyed upon termination. In some embodiments, the policy for filtering events into members of set of storage objects 1400 is based on an identifier of the browser, of the user, an identifier of a partner relationship, an identifier of a compensation arrangement; an identifier
40 of other policies could also be used. In some embodiments the identifier acts as a key in a key-value lookup data structure.

45 An aggregation engine 1500 requests the SLSE 1300 from the set of storage objects 1400. Using a set of rules 1600 the aggregation engine 1500 creates enterprise facts 1700 (hereinafter "EF"). Next, a second message broker 1800 receives the EF 1700 for further
50 distribution.

Fig. 2 depicts architectural components in Fig. 1 in greater detail. A storage object 2100 is an example of one of the set of storage objects 1400 in Fig. 1 and stores SLSE. The storage object includes a data structure 2150 for storing SLSE. The data structure 2150 should be suitably chosen based on the type of filtering criteria used to allocating SLSE across plural storage objects 2100. In some embodiments, SLSE are allocated based on a session creating the SLSE and the data structure 2150 is a hash table that maps keys to object values where the key is the type of event and object value the event. Other data structures available to one skilled in the art could be used depending on the needs of the situation.

The storage object 2100 contains several interface methods described in the following table:

Table 1.	
Method	Function
void init()	Creates an instance of the Storage Object
void destroy()	Destroys the Storage Object
void storeEvent(Event anEvent)	Stores an incoming event in the Storage Object
Boolean containsEvent(Key aKey)	Tests if the Storage Object contains an event
Event retrieveEvent(Key aKey)	Retrieves an event from Storage Object
int removeEvent(Key aKey)	Removes an event from Storage Object
EventCollection getEventCollection()	Returns a collection of the events stored in the Storage Object

The aggregation engine 1500 generates EFs. EFs are generated through a process of satisfaction of conditions in rules. The conditions are associated with the SLSEs; in some embodiments they are SLSE's themselves, in others they could be identifiers of SLSEs, or a transformed representation. When conditions in the rule are satisfied, an EF generated. The aggregation engine 1500 includes a rule data structure 1550 that contains a plurality of rule objects 1600. The rule objects 1600 comprise a trigger event 1620 and, optionally, other events 1610. The rule objects 1600 also includes an embedded action object 1630. It is not fundamental that the action object 1630 be embedded and in some embodiments, it is not embedded although still accessible from the storage object 1400. As described in more detail below, the aggregation engine 1500 executes the action object 1630 when the SLSE corresponding to the trigger event 1620 arrives.

Now, in greater detail, a rule 1650 includes a condition portion 1660 and an action portion 1705. The condition portion 1660 comprises a trigger condition and zero or more additional conditions. In some embodiments the conditions are satisfied by corresponding SLSE

1300. In some embodiments, the action portion of the rule 1650 is triggered if, and only if, the event corresponding to the trigger condition has arrived; in other embodiments the rule could be triggered if a more complex pattern of events or conditions is satisfied.

The action 1705 includes a base action 1760 and a complex action 1770. In some embodiments, the complex action 1770 is optional. The base action 1760 includes an identifier of the EF to generate when executing the base action 1760.

The complex action 1770 provides for the creation of EFs that reflect, or are based on, a function of SLSEs. The complex action 1770 allows for aspects of several SLSEs, perhaps distant in time, to contribute to an EF and, further, allows only the relevant aspects of the contributing SLSEs to be stored, thus freeing an implementing system from the storage burden of several SLSEs.

By way of example, consider a scenario in which a portal-type website implements a common-basket shopping model, an EF may be a Total Purchase Amount during a particular browsing session. This may be represented by a sum of the individual purchases (also an EF) for that session which is derived from SLSEs, say following a particular link (the purchase action). In this scenario, it is desired that the individual purchases EF accumulate the purchase prices throughout the session, and when the session completes (by logout, timeout, etc.) then the EF indicating total purchases for this session is produced. Other examples include, for instance, counting pages viewed, counting a number of times a given page is viewed, statistics relating to navigation paths, statistics relating to virtual "shopping carts/baskets" such as minimum/maximum items (or amount) or final state.

The complex action 1770 allows for this type of behavior (as well as much more general complex behaviors). The complex action 1770 includes a function 1710 to perform on a EF 1720, a field 1730 in the EF 1720 to affect, a SLSE 1740 for the function 1710 to use and a field in the SLSE 1750 to take as input.

In operation, the scenario could operate by having a first rule for generating the EF of an individual purchase—the base action 1760. The function 1710 could be an accumulator and the Total Purchase Amount EF could be the EF 1720 that is accumulated by the function 1710. The

field 1730 would be an amount field to accumulate. The SLSE 1740 and the field in the SLSE 1750 would be the appropriate field in the one of the conditions 1660 to add—namely the purchase price of the individual purchase. One of skill in the art, having the benefit of this disclosure will appreciate that this framework for the rule 1650 affords great flexibility in generating EFs. Still further, it frees an implementing system from storing the SLSEs and rather allows for the efficient storing of just the accumulated total of their contribution to the EF of total session purchase amount. As one skilled in the field will by now recognize, operation of the complex action 1770 is not limited to the above scenario. Rather, one skilled in the art, having the benefit of this disclosure will recognize other situations where this aspect could be beneficially employed.

Returning to the aggregation engine 1500, the illustrative embodiment includes several interface methods set forth in the following table:

Table 2.	
Method	Function
AggregationRule(String aRule)	Constructor for a Rule from a String
AggregationRule()	Default constructor for a Rule
int getTrigger()	Returns a code of the trigger event
void setTrigger()	Sets a trigger event for the rule
void setCondition(String aCondition)	Set a condition for the rule
void setConditions(String Conditions)	Set all the conditions of the rule
void setAction(String anAction)	Set the action will be executed by the rule
Enumeration getConditionEnumeration()	Returns an enumeration of the conditions which compose the rule
void release()	Release resources acquired by the rule, specifically by its action object
EF executeAction(EventCollection aEventCollection)	Execute the action

Many types of EFs can be implemented. Architecturally, an EF base class 1810 that includes attributes common to EFs is used. The common attributes may vary depending on the situation, although typical attributes such as a creation date timestamp and a session identifier for the creating session are preferably included. A set of derived EF classes 1820 which include the particular attributes and methods unique to the various EFs desired by a system operator inherent from the EF base class 1810.

Fig. 3 depicts a flow diagram of an operational scenario for efficient generation of EF from SLSE in accordance with an illustrative embodiment. Process flow initiates when the aggregation engine 1500 invokes the getEvent method 4100 on one of the set of storage objects

1400 and receives a SLSE 4150 in return. The aggregation engine 1500 maps 4200 the SLSE to the rule objects for which the SLSE 4150 is a condition. In some embodiments, SLSEs have an associated identifier and an efficient data structure, e.g., a hash table, is used for the mapping where the associated identifier maps to rules for which the SLSE occurs in the condition portion 1660. The aggregation engine 1500 sets 4250 the SLSE 4150 and tests whether the trigger event is satisfied so that the rule object 1600 should to execute the action 1630.

Assuming for illustration, the trigger event is not satisfied, process flow returns to the aggregation engine 1500 that invokes the `getEvent` method for a new event 3155 that again returns an SLSE 4300 which is mapped 3205 to a rule object 1600. Assuming, now, that the SLSE 4300 is a triggering event, the aggregation engine 1500 sets 3255 the trigger condition 1620. The trigger condition 1620 being satisfied 3400, the aggregation engine 1500 invokes the `executeAction` method 3450 of the action object 1630 in the rule object 1600. The action object 1630 then executes the `create` method 3500 for the EF 1700 which is passed to the second message broker 1800 for further distribution.

Distribution through the message broker 1880 provides additional benefits in connection with features of the invention. For instance, in an optimized system for transforming SLSEs to EFs where EFs may be used by several entities, distribution through the message broker 1880 allows for a system operator to reduce storage requirements by simply publishing the EFs via the message broker 1880.

Fig. 4 depicts a block diagram of an operating architecture for efficient generation of enterprise facts ("EF"s) from browsing session event data in accordance with an embodiment distinct from that described in connection with Fig. 1. Certain features are common, as will be appreciated by one skilled in the field from this disclosure. As previously described in connection with Fig. 1, SLSE 1300 arrive from the first message broker 1100. In this embodiment, an accumulation container manager 4100 receives the SLSE 1300. The accumulation container manager 4100 regulates the SLSE into accumulation containers 4300 with a container manager policy 4200. The container manager policy 4200 provides a policy for dispatching SLSE 1300 into containers 4300. No policy is fundamental and the policy should be selected based on the requirements of the situation. Illustrative policies include, for instance,

based on a session of a user, based on the user, based on a product or service, based on a business agreement, based on other quantities, or based on the types the SLSE 1300. In an illustrative embodiment, the container manager policy 4200 includes an interface method that returns the association between the containers 4300 and a particular one of the SLSE 1300, an interface method for shutdown and restart as described below, as well as a destructor method and a method for deleting container-identifying keys.

An accumulation process accessing the container 4300 performs the function of aggregating the SLSE 1300. In accordance with the embodiment in Fig. 4, the container 4300 simply has access to the rules object 4600 and the action object 4630. For each container 4300, as the SLSE 1300 filter in, the accumulation process identifies each of the SLSE 1300 by type. Based on the type, the aggregation process then determines the particular ones of the rule objects 4600 and the action objects 4630 that should receive each of the SLSE 1300. It should be made clear that each one of the SLSE 1300 could be mapped to more than one of the rules objects 4600 and the action objects 4630 if appropriate based on the type.

The rule object 4600 performs whatever aggregation, accumulation, or operation that is desired based on the SLSE 1300. More particularly, a predefined rule performs some function based on the data from the SLSE 1300. What the particular function is will vary with the needs of the situation. This could be, for instance, a simple counter, e.g. counting number of page views in a user session, counting revenue accumulated by a merchant during a user's shopping session. This could also be more complex formulation, for instance, based on a number of conditions or cases. For ease of exposition, the term "Accumulation" is used hereafter to refer to functions performed by the rule objects 4600. Conveniently, when plural rule objects 4600 need to perform the same Accumulation, this Accumulation could be factored out into an intermediary rule object 4600 that the others could reference.

Analogously, the action object 4630 performs the action is based on the SLSE 1300. The Accumulation process determines the type of each of the SLSE 1300 which is used to identify one (or more) of the action objects 4630. The action objects 4630 receive the SLSE 1300 that is a triggering conditions for executing whatever action the particular action object 4630 performs. This action includes the production of an EF; in some instances it could be the production of an

intermediary result.

5 In accordance with this embodiment the accumulation process accesses the containers 4300 and for the SLSE 1300 tests whether each SLSE 1300 is a “parameter” for the rule objects 4600—that is whether the rule object 4600 performs an Accumulation based on SLSE 1300 of
10 that type. In addition, the accumulation process tests whether each SLSE 1300 is a “condition” for one of the action objects 4630. That is, the rule objects 4600 identify when one of the SLSE 1300 should trigger an action and the particular action object 4630 for execution.

15 When the action objects 4630 generate EFs, the EFs pass to a consolidator 4400. The consolidator 4400 allows for synchronization of the flow of EFs with a persistent storage. In some embodiments, an object repository 4800 provides a persistent object storage for objects
20 which are relevant to the enterprise. For instance, so-called “business objects” are an example of such a set of objects relevant to an enterprise and the object repository 4800 could be a storage for such business objects. The particular storage architecture is not fundamental. In some
25 embodiments, a relational database management system is used. With an XML-based object representation used elsewhere, an object-relationship mapping could be used to provide an abstraction layer between the XML and relational models. A dedicated object manager could
30 also be used, as could other storage models available to one skilled in the field. The consolidator 4400 interoperates with the object repository 4800 so that downstream users of the EFs are able to obtain consistent, synchronized, information whether through access of the object repository
35 4800 or from the EFs. In some embodiments, an interpretation of the production of EFs is as a state change for business objects stored in the object repository 4800.

40 When an EF is generated, the consolidator 4400 receives the EF and takes any action necessary to retain consistency with the object repository 4800. The consolidator 4400 generally provides a transactional-based access to external resources in connection with producing EFs and maintaining consistency with the object repository 4800. For instance, the consolidator 4400
45 could retrieve or updating additional information in connection with producing EFs. The consolidator 4400 provides the EF to a consumer manager 4500 for further distribution via the second message broker 1800. One feature the consolidator 4400 can implement is simultaneous
50 commit of both the distribution to the consumer manager 4500 and the consistency maintenance

55

against the object repository 4800.

5 The consumer manager 4500 controls distribution of EFs to recipients of the EFs via the second message broker 1800. Reliable transmission protocols could be used, and the consumer manager 4500 could ensure reliable distribution of the EFs to clients receiving from the second
10 message broker 1800.

 Fig. 5 depicts a flow diagram of an illustrative operational scenario for efficient generation of EF from SLSE in accordance with embodiments illustrated by Fig. 4. Upon arrival
15 of a first event 5010, the accumulation container manager 4100 invokes a getContainer method 5200 in the container policy manger 4200 for determining to which container 4300 the first event 5010 should be dispatched and the container policy manager 4200 returns 5030 the appropriate
20 container. No particular manner of identifying the container is fundamental, for instance a key could be passed for use with a look-up data structure, object reference, or other means. If a container needs to be created, for instance if the first event 5010 indicated the initiation of a session and containers were session based, then the accumulation container manager 4100 would
25 create a new container.

 The accumulation container manager 4100 invokes a method 5040 in the container 4300
30 with the first event 5010 and the accumulation process tests 5050 the rule object 4600 based on the type of the first event 5010. If the first event 5010 is of the type the rule object 4600 accumulates, the rule object 4600 returns 5060 this indication. The rule object 4600 also
35 accumulates 5055 one or more quantities it tracks based on the first event 5010. The accumulation process also tests 5070 whether the first event 5010 is a condition for one of the action objects 4630. In this illustrative scenario, a false is returned 5080.
40

 In this illustrative scenario, when a second event 5085 arrives, the accumulation container manager 4100 again consults 5090 the container manager policy 4200 which returns 5100 the
45 proper container 4300 for the second event 5085.

 The accumulation container manager 4100 invokes a method 5110 in the container 4300
50 with the second event 5085 and the accumulation process tests 5120 the rule object 4600 based on the type of the second event 5085. If the second event 5085 is of the type the rule object 4600
55

accumulates, the rule object 4600 returns 5130 this indication. The rule object 4600 accumulates
5 5135 one or more quantities it tracks based on the second event 5085. Now, illustratively, the
second event 5085 is also identified by the accumulation process as associated with the action
object 4630; this is tested 5140 and the action object 4630 returns 5160 returns this indication.
An execute method is invoked 5170 in the action object 4630 to create 5180 the enterprise fact
10 1700. As has been previously described, the enterprise fact 1700 can be based on quantities
accumulated in one or more rule objects 4600. The enterprise fact 1700 is passed back 5190 to
the container 4300 and then passed 5200 for further processing, for instance by the consolidator
15 4400.

While it should be apparent to one skilled in the field, it is noted that the first event 5010
20 and the second event 5085 discussed above in connection with Fig. 5, are examples of the SLSE
1300.

Additional features are noted in connection with the aforementioned illustrative
25 embodiments. First, stop and restart functionality could be implemented in connection with the
system and method for transforming session data described herein. In such an instance the state
of the containers 4300 and/or the storage object 1400 should be saved at stop time (as well as any
30 of the SLSE 1300 being currently processed) and the state resumed at restart time. Second, it
may be desirable to further implement "failover" functionality so that state can be resumed in
case of inadvertent stopping of processing, e.g., system crash, power failure, etc. One skilled in
35 the art having the benefit of this disclosure will appreciate how known techniques could be
applied in this regard.

Although the present invention has been described in terms of features illustrative
40 embodiments, one skilled in the art will understand that various modifications and alterations
may be made without departing from the scope of the invention. Accordingly, the scope of the
invention is not to be limited to the particular embodiments discussed herein, but should be
45 defined only by the allowed claims and equivalents thereof.

Claims

What is claimed is:

5

1. A computer-controlled method of generating enterprise facts from session-level semantic events comprising:

10

generating session-level events responsive to data flow of a browsing session of a user;

receiving and accumulating said session-level semantic events in a storage object;

15

determining if one of said session-level semantic events is a triggering condition of one or more rules for generating high-level semantic; and

returning a enterprise fact for those of said one or more rules for which said triggering condition is satisfied.

20

25

2. The computer-controlled method according to claim 1 wherein said storage object is associated with said browsing session.

30

3. The computer-controlled method according to claim 1 wherein said storage object is associated with said user.

35

4. The computer-controlled method according to any preceding claim where generating session-level events also includes storing said session-level events in a shared memory area.

40

45

5. The computer-controlled method according to any preceding claim wherein returning a high level semantic event includes broadcasting across a message broker.

50

55

- 5 6. The computer-controlled method according to any preceding claim wherein said one or more rules comprise:
- 10 a condition portion, said condition portion comprising a first event portion and a second event portion, said first event portion associated with a triggering event, said second event portion associated with other events.
- 15 7. The computer-controlled method according to any preceding claim wherein said session-level events are a data object including a set of predetermined attributes and a set of pair-value attributes.
- 20 8. The computer-controlled method according to any preceding claim wherein said one or more rules comprise:
- 25 a first action part, said action part comprising, a default action indicating a first type of enterprise fact to return, and a second action part indicating a function to be performed on second type of enterprise fact.
- 30 9. The computer-controlled method according to claim 8 wherein said second action part comprises:
- 35 a function type;
- 40 a enterprise fact type to be affected; and
- 45 a field of said enterprise fact to be affected.
- 50 10. The computer-controlled method according to any preceding claim where there is a many to one mapping between said generated session-level semantic events and
- 55

said returned enterprise facts.

5

11. A computer-implemented system for generating enterprise facts from low-level semantic events comprising:
- 10 a storage object, said storage object having a method for storing a low-level semantic event, and a method for testing if said low-level semantic event is stored;
- 15 a enterprise fact generation object comprising a method for invoking said method testing if said low-level semantic event is stored in said storage object for determining if a condition of a predetermined rule is satisfied;
- 20 a rule object comprising a set of conditions including a trigger condition for activating a rule; and
- 25 an action object for execution upon activation of said rule.

- 30 12. The system according to claim 11 wherein:
- wherein said enterprise fact generation object comprises a data structure for storing a value associated with a key, and wherein said key is said trigger condition, and
- 35 said value is said rule.

- 40 13. The system according to claim 11 or claim 12 further comprising a first message broker, said message broker configured for receiving enterprise facts from said high enterprise fact object.
- 45

14. The system according to any of claims 11 to 13 wherein said one or more rules
- 50 comprise:

55

a plurality of triggering conditions;

an first action part, said action part comprising, a default action indicting a first type of enterprise fact to return, and a second action part indicating a function to be performed on second type of enterprise fact.

15. The system according to any of claims 11 to 14 wherein said one or more rules comprise:

a function type;

a enterprise fact type to be affected; and

a field of said enterprise fact to be affected.

16. The system according to claim 15 wherein said one or more rules comprise:

an first action part, said action part comprising, a default action indicting a first type of enterprise fact for returning, and a second action part indicating a function to be performed on second type of enterprise fact.

17. A computing apparatus comprising:

means for receiving session-level semantic event data, from a user's session;

means for accumulating said event data;

means for determining if one of said session-level semantic event datum is a triggering condition of one or more rules for generating enterprise fact; and

means for returning a enterprise fact for those of said one or more rules for which said triggering condition is satisfied.

5

18. A computer-implemented method for transforming user-session data into business event data, said method comprising:

10

receiving events created from a user session;

15

storing a plurality of rules, said rules having a condition part and an action part, said condition part being satisfied responsive to selected ones of said events from said user session, said action part for creating business event data;

20

matching said events created from a user session with ones of said rules for which said events contribute to satisfaction of the condition part;

25

generating business event data through executing the action part of said rules upon determining the condition part of said rules is satisfied; and publishing said business event data on a message broker.

30

19. The method of any of claims 1 to 10 wherein said storage object is non-persistent.

35

20. The system of any of claims 11 to 16 wherein said storage object is non-persistent.

40

45

50

55

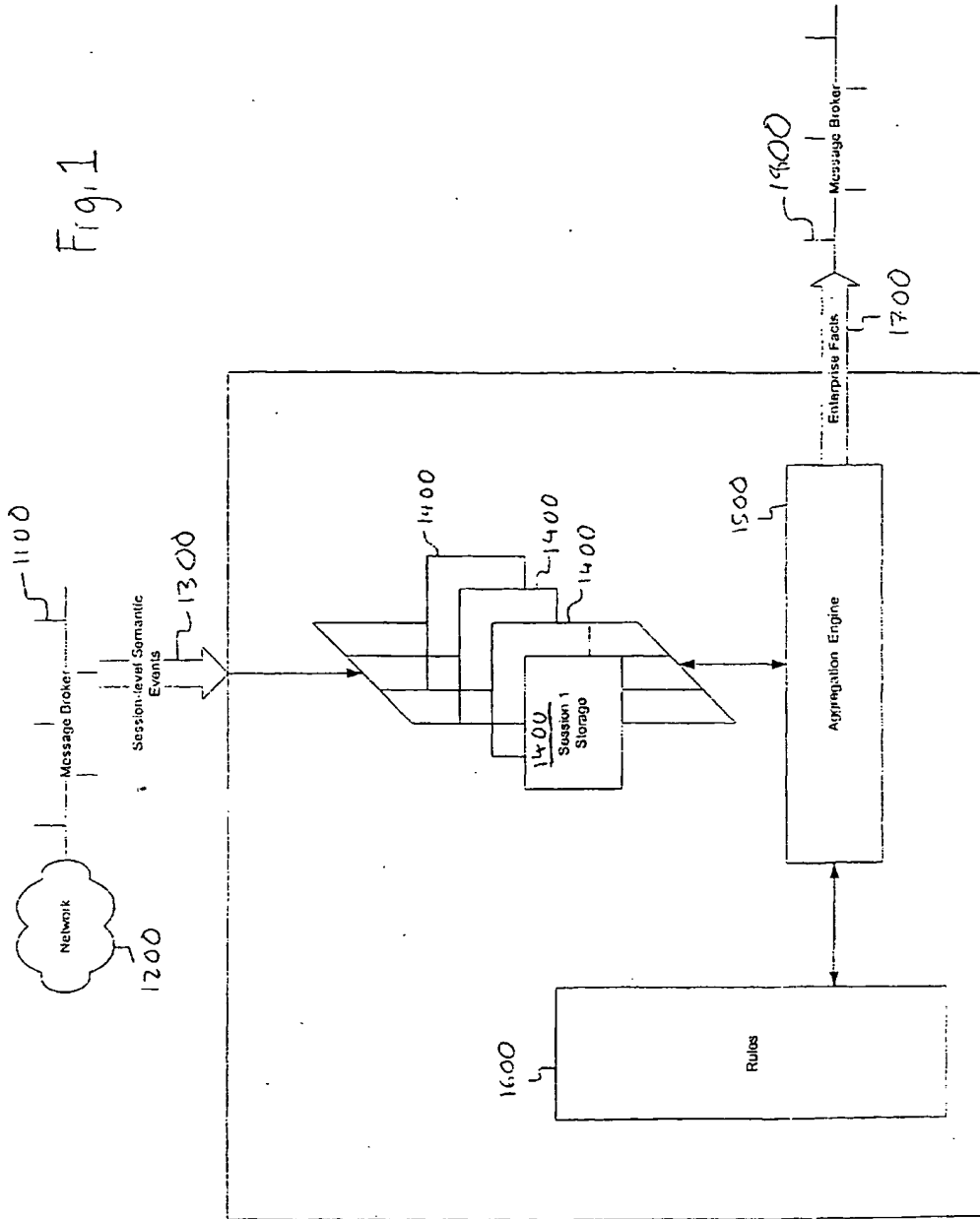
ABSTRACT OF THE DISCLOSURE**METHOD AND SYSTEM FOR TRANSFORMING SESSION DATA**

A disclosed method for transforming data from client-server request /response cycle includes receiving session-level semantic events. The session-level semantic events are mapped to accumulation containers based on a policy, for instance, based on a session identifier. The session level semantic events are examined for a type. From the type, the session level semantic events are mapped to one or more rules which perform aggregations, accumulations, or other operations based on the data from the session level semantic events. The rules perform these aggregations, accumulations, or other operations over a set of session level semantic events for generating information (termed "enterprise facts") that are independent of the client-server protocol and have an meaning in context that spans more than the request/response cycle. Enterprise facts are generated when an session level semantic event that is a triggering condition for an action is received. When the enterprise fact is generated it can draw on the aggregations, accumulations, or other operations performed by the rules in that accumulation container. Also disclosed is the use of a message broker for receiving the session level semantic events from their source and the use of message broker for distributing the enterprise facts.

Fig 1

5
10
15
20
25
30
35
40
45
50
55

Fig. 1



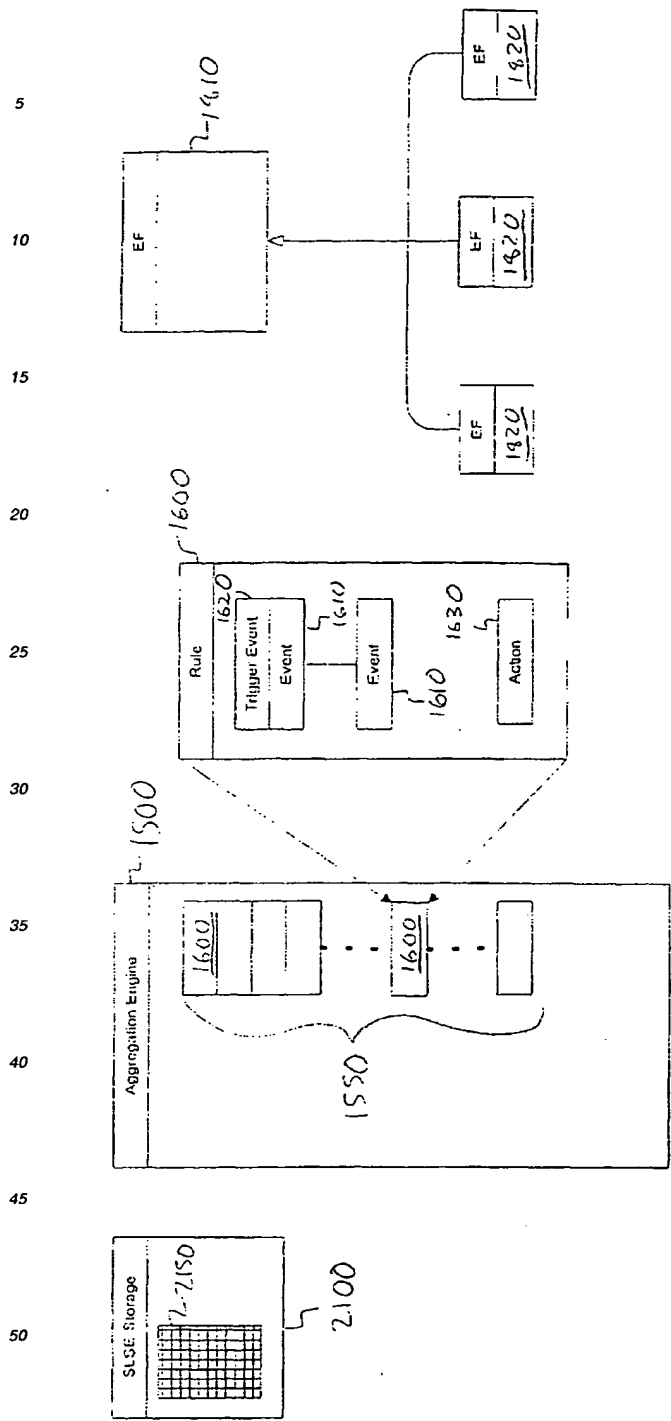
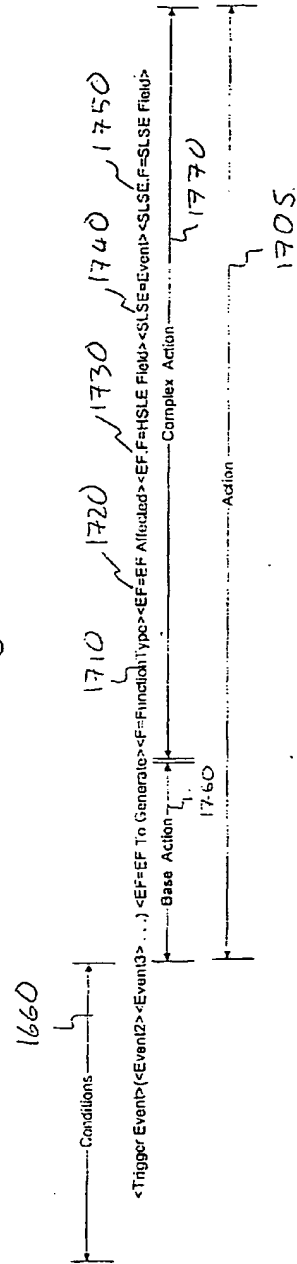
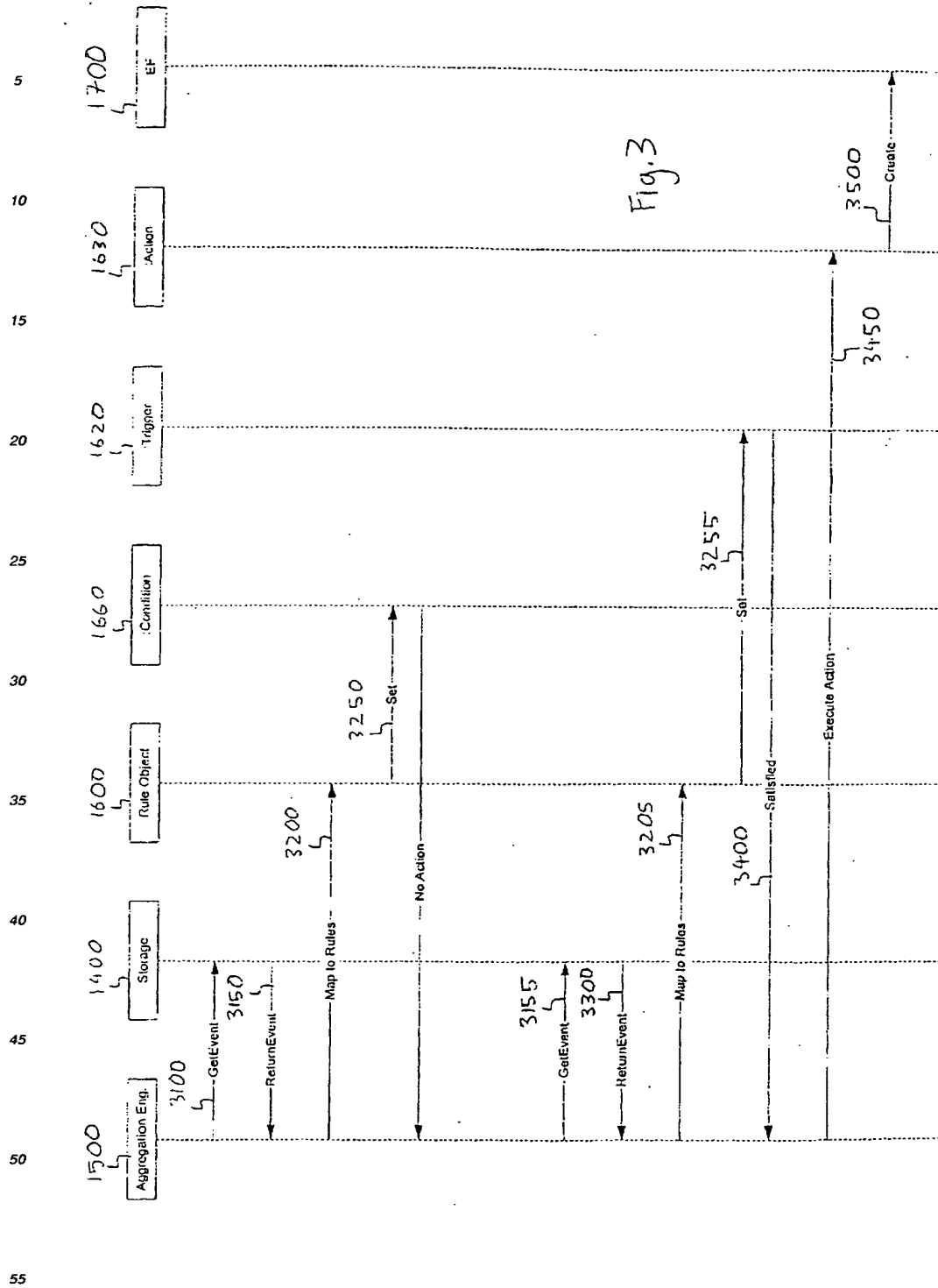


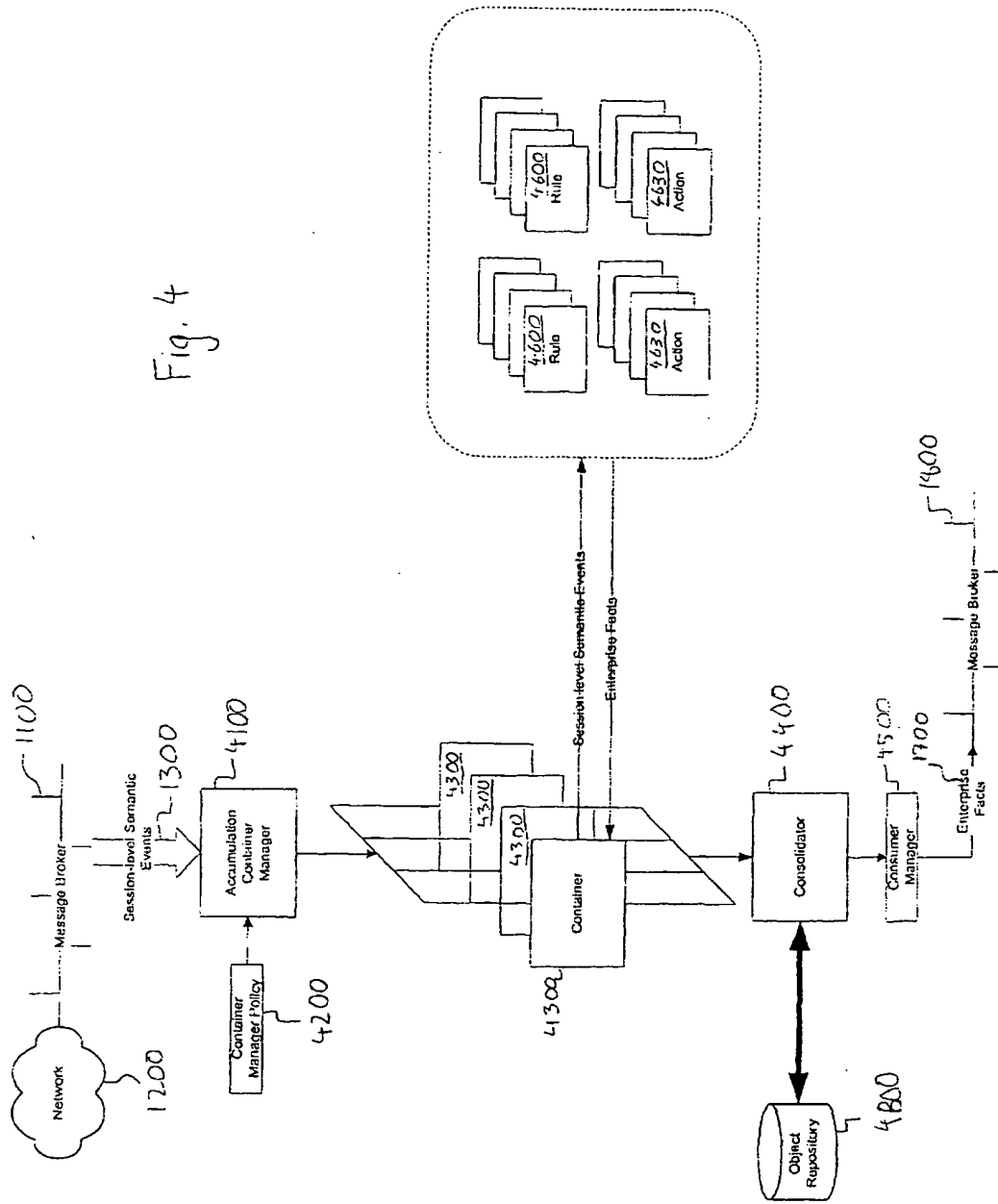
Fig 2

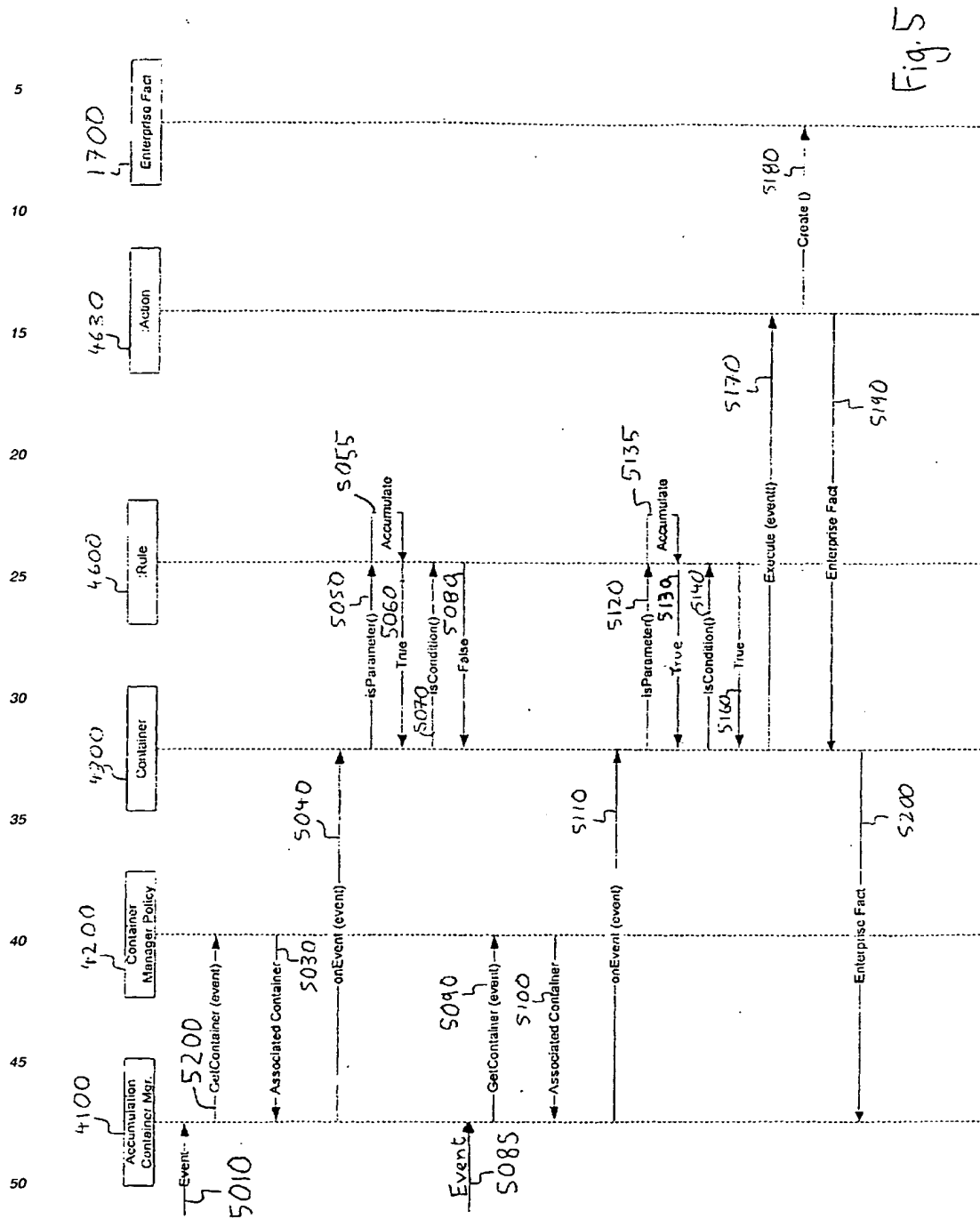




5
10
15
20
25
30
35
40
45
50
55

Fig. 4





Claims

1. A computer-controlled method of operating a network-based partner relationship, said computer-controlled method comprising:

EP 1 189 161 A1

establishing a representation of a relationship between a first peer entity and a second peer entity, said representation comprising: a collection of resources associated with said relationship, a presentation representation to be applied when providing ones of said collection of resources, and criteria for providing a compensation flow between said first peer entity and said second peer entity;

5

dynamically generating composite resources during a browsing session, said composite resources comprising a first component selected from said collection of resources and associated with said first peer entity, and a second component selected from said collection of resources and associated with said second peer entity;

10

monitoring said browsing session for generating session events;

transforming said session events to events in a semantic associated with said relationship, generating higher-level semantic events;

15

generating a compensation flow responsive to said higher-level semantic events based on said criteria.

2. The computer-controlled method according to claim 1 wherein:

20

generating session events further comprises providing said session events on a first message broker; and

wherein transforming session events comprises:

receiving said session events from said message broker; and

25

providing said higher-level semantic events on a second message broker.

3. The computer-controlled method according to claim 2 wherein dynamically generating composite resources during a browsing session comprises:

30

receiving said higher-level semantic events from said second message broker; and

selecting one of said first component and said second component responsive to said higher-level semantic events.

35

4. The computer-controlled method according to any preceding claim wherein dynamically generating composite resources during a browsing session comprises:

initiating a request for said first component, said request directed to a first target site;

40

receiving said request;

altering said request, forming an altered request; and

forwarding said altered request to said first target site.

45

5. The computer-controlled method according to any preceding claim wherein dynamically generating composite resources during a browsing session comprises:

initiating a request for said first component, said request directed to a first target site;

50

receiving a response from said first target site;

altering said response forming an altered response;

55

forwarding said altered response; and

receiving said altered response.

6. The computer-controlled method according to claim 4 further comprising:
- initiating a request for said first component, said request directed to a first target site;
- 5 receiving said request;
- extracting information from said request;
- altering a state of a state object responsive to said information extracted from said request.
- 10 7. The computer-controlled method according to claim 6 said state object comprises contents of a common shopping basket.
8. The computer-controlled method according to claim 6 wherein said state object comprises a representation of value stored of a common wallet.
- 15 9. A computer-controlled network-based partnership system comprising:
- a relationship engine for defining attributes of a relationship among peer entities comprising a first peer entity and a second peer entity;
- 20 a storage for storing said attributes of said relationship;
- a composite site manager for dynamically generating composite resources during browsing sessions, said composite resources comprising component resources associated with said first and second peer entities, said composite site manager configured for retrieving a characterization of said composite resources from said storage;
- 25 a session tracking component, said session tracking component comprising:
- a server application configured for receiving requests from a client system,
- a communication client configured for requesting said composite resources from said composite site manager,
- 35 a client application configured to request said component resources from server systems associated with said first and second peer entities; and
- a session event generation module configured for generating event data respecting said requests and said component resources;
- 40 a set of semantic mapping modules, said semantic mapping modules configured for receiving , said semantic mapping modules comprising rules for generating higher-level semantic events responsive to said event data.
- 45 10. The system according to claim 9 further comprising:
- a message broker, said message broker configured for receiving said higher-level semantic events from said set of semantic mapping modules; and
- 50 wherein said composite site manager is further configured for receiving said higher level semantic events from said message broker, and wherein said composite site manager dynamically generates said composite resources responsive to said higher-level semantic events.
11. The system according to claim 9 or claim 10 further comprising:
- 55 an interposition module, disposed between said client application and said server systems, said interposition module configured for altering said request sent by said client application for said component resources.

12. The system according to claim 9 or claim 10 further comprising:

an interposition module, disposed between said client application and said server systems, said interposition module configured for altering said component resources sent by said server systems.

5

10

15

20

25

30

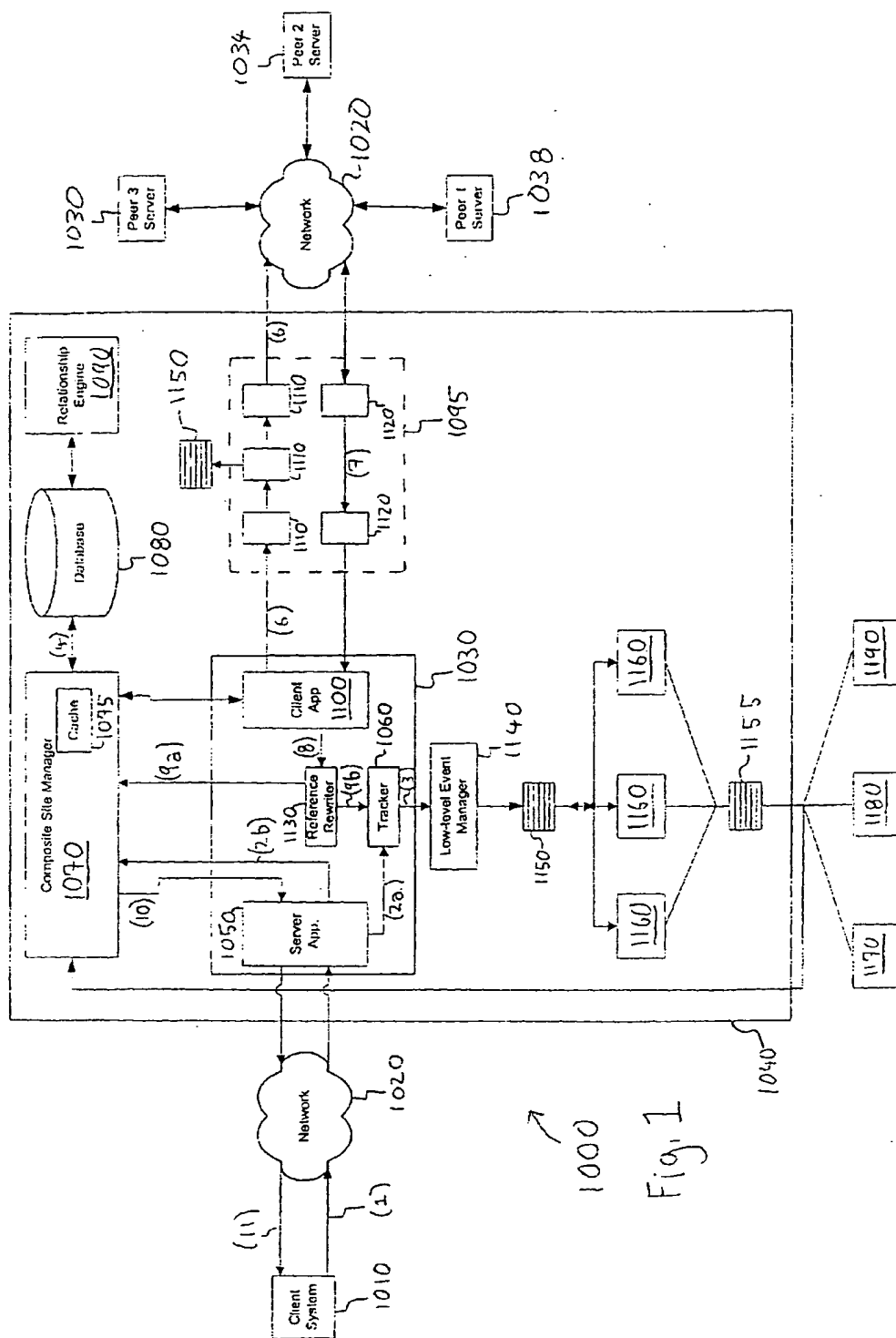
35

40

45

50

55



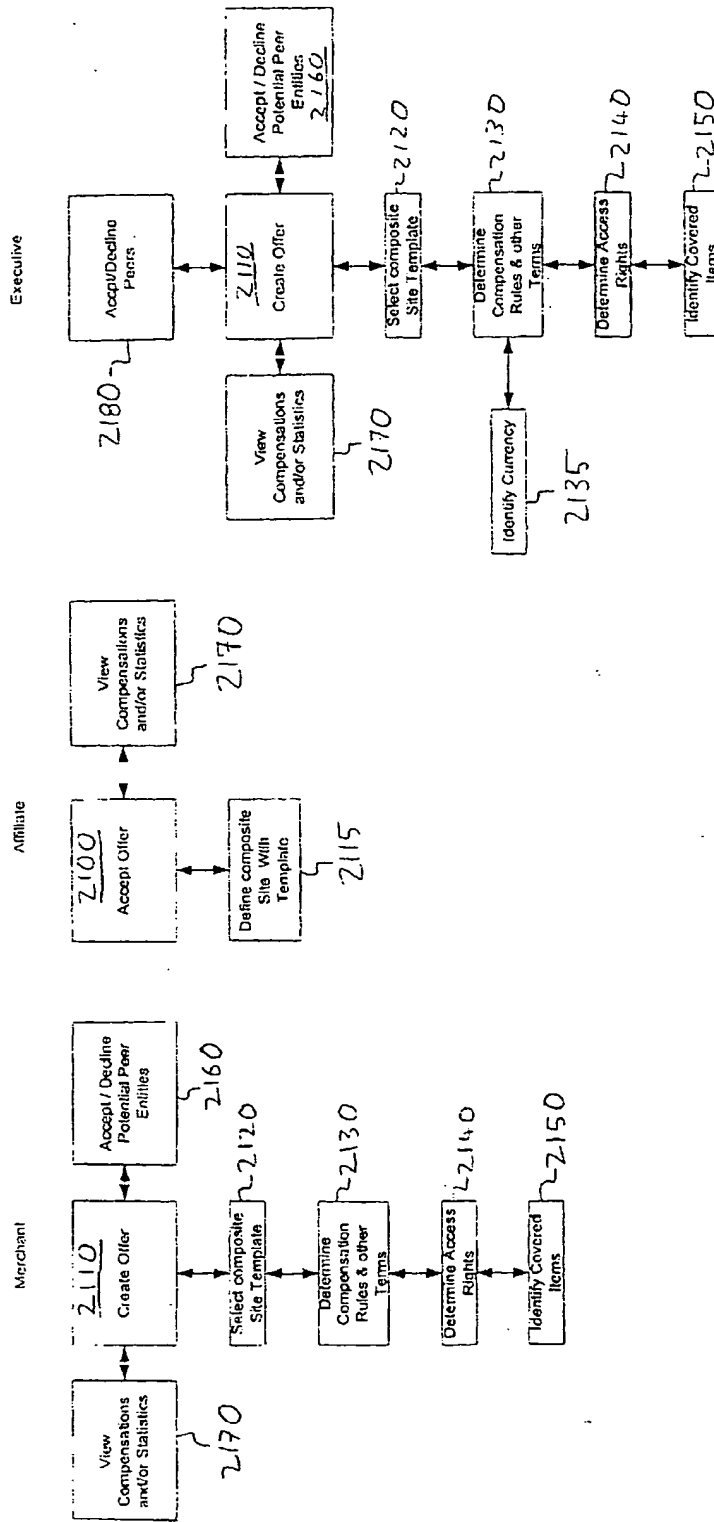


Fig 2-1

Fig.2-2

Fig.2-3

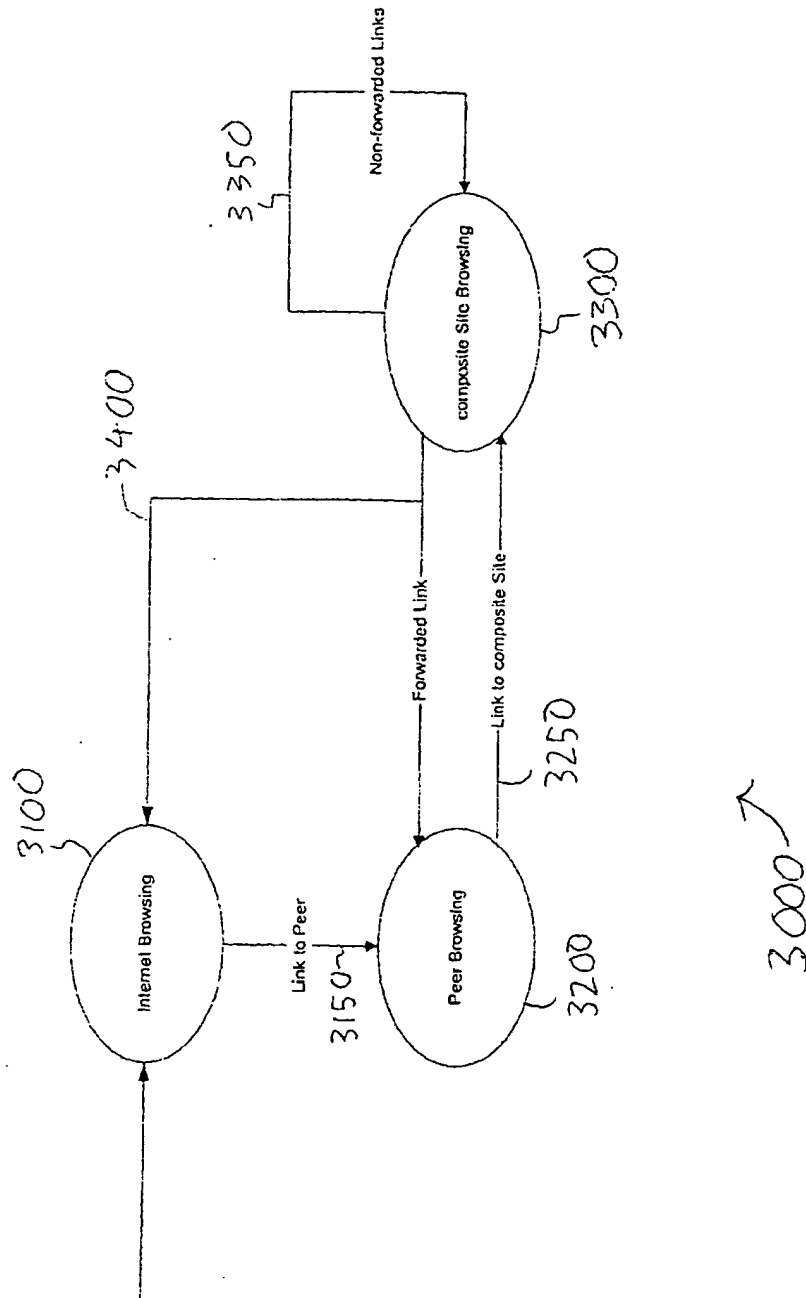
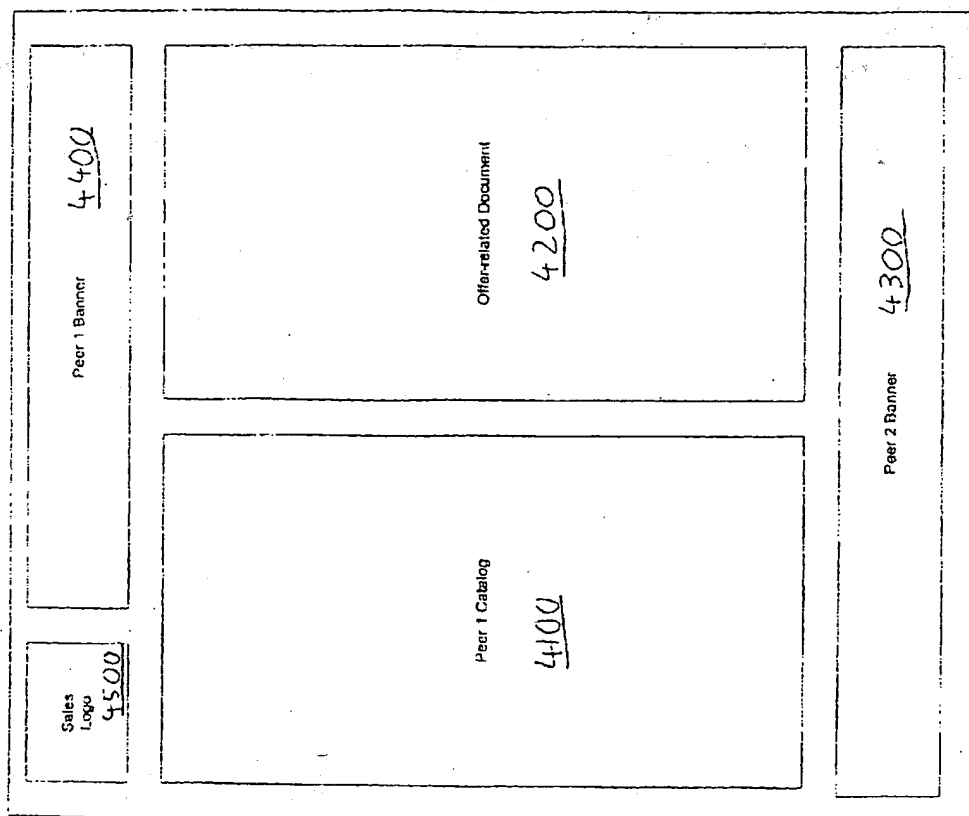


Fig. 3



4000 →

Fig. 4

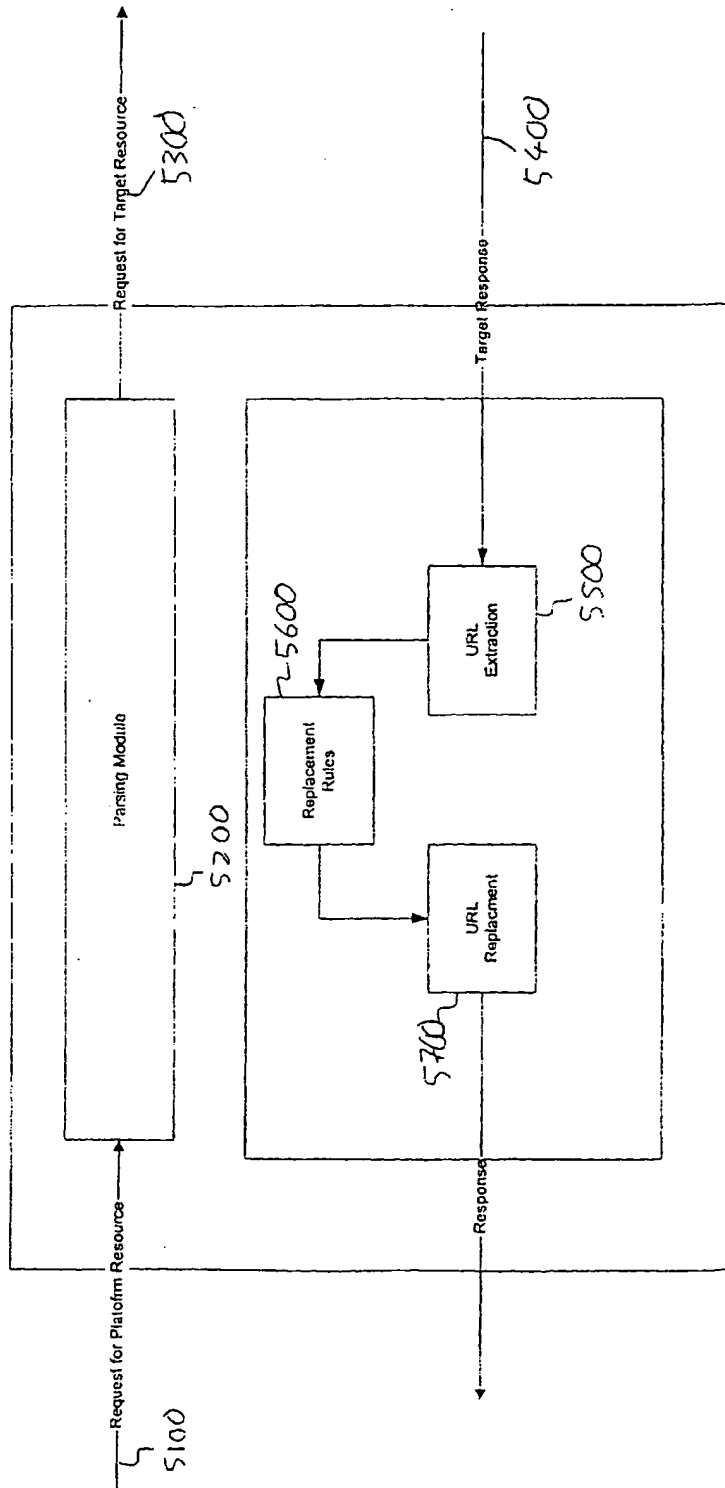


Fig. 5



European Patent
Office

EUROPEAN SEARCH REPORT

Application Number
EP 00 40 2519

DOCUMENTS CONSIDERED TO BE RELEVANT			
Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.7)
Y	US 5 812 769 A (WEINBERGER MARVIN I ET AL) 22 September 1998 (1998-09-22) * abstract * * figure 1 *	1-12	G06F17/60
Y	US 5 948 061 A (MERRIMAN DWIGHT ALLEN ET AL) 7 September 1999 (1999-09-07) * abstract * * figure 1 *	1-12	
A	WO 00 14665 A (HAKIM PAUL D ;OWNX INC (US); PALCIC PATRIC M (US)) 16 March 2000 (2000-03-16) * abstract *	1-12	
			TECHNICAL FIELDS SEARCHED (Int.Cl.7)
			G06F
The present search report has been drawn up for all claims			
Place of search THE HAGUE		Date of completion of the search 10 July 2001	Examiner Falierou, C
CATEGORY OF CITED DOCUMENTS		I : theory or principle underlying the invention E : earlier patent document, but published on, or after the filing date D : document cited in the application L : document cited for other reasons & : member of the same patent family, corresponding document	
X : particularly relevant if taken alone Y : particularly relevant if combined with another document of the same category A : technological background O : non-written disclosure P : intermediate document			

EPO FORM 1503 03 82 (P2-C/1)

**ANNEX TO THE EUROPEAN SEARCH REPORT
ON EUROPEAN PATENT APPLICATION NO.**

EP 00 40 2519

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

10-07-2001

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5812769 A	22-09-1998	AU 7240396 A WO 9711429 A	09-04-1997 27-03-1997
US 5948061 A	07-09-1999	NONE	
WO 0014665 A	16-03-2000	AU 2251300 A	27-03-2000

EPO FORM P0189

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82